RISK IN THE DEVELOPMENT OF DESIGN

Rose Crossland

A thesis submitted to the University of Bristol in accordance with the requirements, in application for the degree of Doctor of Philosophy in the Faculty of Engineering, Department of Engineering Mathematics.

August, 1997

ABSTRACT

There is a need in engineering enterprises for methodologies and tools to support design engineers and their managers in decision-making during the early stages of the design process - a time when both the structure and the attributes of the designed artefact are highly uncertain and hence much risk is present. This research work is centred on a risk modelling methodology, and supporting software tool, which have been developed to enable teams of designers to build a shared **risk model** - a hierarchical model of the design which incorporates uncertainty, is capable of incremental refinement to reflect the development of the design during the design process and which may be evaluated to yield a risk assessment for the design project.

A class-based object-oriented modelling methodology has been extended to represent probabilistic uncertainty in attribute values, in the values returned by algorithms and in the relationships between objects. The methodology also allows multiple methods to be defined for the calculation of an attribute value, with more accurate methods being automatically invoked as their input information becomes available. A technique termed **hierarchical risk sensitivity analysis** has been developed which may be used to identify the major sources of uncertainty at any level in a hierarchical decomposition of the risk model. Support for **configuration modelling under uncertainty**, involving the inclusion of multiple design variants at any level in the hierarchy, has also been incorporated into the methodology. The software tool (**RiTo**) was used to evaluate the methodology on case studies conducted with two collaborating industrial companies. Specific design projects were used to test whether the tool and methodology could be applied to two contrasting industries.

The research demonstrates, in diverse design domains, the feasibility of the methodology as a means of incorporating uncertain information into early design models. The structures and representations of uncertainty offered by the risk model were shown to meet the modelling needs of the case studies. Automated tools can use such a risk model to provide simple summary risk measures whose development over time supports design decision making, and the research demonstrates suitable algorithms. A methodology and software tool have been developed which have wide applications.

ACKNOWLEDGEMENTS

I would like to offer my truly grateful thanks to my research supervisors, Mr Jon H. Sims Williams and Mr Chris A. McMahon, for providing me with the opportunity to undertake this project and for all their help, guidance and encouragement throughout the work. I could not have asked for more patient or helpful supervision.

This work was carried out as part of the STARTED project (grant reference GR/J53867) which was supported by the DTI / EPSRC CSCW programme, and I am grateful to them for their support. I am also grateful to the industrial partners in the STARTED project - Rover Group Ltd, Cegelec Projects Ltd, and Ove Arup and Partners.

My thanks are due to all those at Rover Group Ltd, Cegelec Projects Ltd and Ove Arup and Partners who gave their time and effort to the project. I would particularly like to thank Dr Neil Davis of Warwick Manufacturing Group who acted as an intermediary with Rover during the the project and who gave much time and effort to the work and made many illuminating and useful suggestions. I must also thank Mr John Todd, Mr Janarrdan Devlukia, Mr Dave Simmonds, Mr Ed Jackson, Ms Lorna MaCaulay and Mr John Raulston, all of Rover Group Ltd, for their contributions. I am especially grateful to Mr Roger Watson and Dr Laurie Burrow of Cegelec Projects Ltd for their interest and support - thanks are due to Mr Watson for his highly efficient, but always kindly, management of the STARTED project. Thanks also to Mr Pat Marriott of Cegelec Projects Ltd. I am grateful to Mr Charles Milloy and Mr Ed Tufton of Ove Arup for fascinating conversations about the nature of risk in their civil engineering projects, and to Mr Tom Barker and Mr Oliver Price, also of Ove Arup.

I would also like to thank all those working at 83 Woodland Road (not only the present occupants, but all those who have worked there over the past three and a half years), for their help and support during this work. Thank you for the many interesting discussions and ideas and thank you for collectively providing a wonderfully friendly and supportive working environment. I am grateful to Dave Brittain for reading a draft version of several chapters of this thesis, and providing many helpful comments (although the mistakes are all my own).

Finally, my deep gratitude, as always, to my partner Dominic for his constant patience and encouragement (and for much appreciated help with corrections and document formatting), and also to my parents Pat and Bill, for everything.

TO DOMINIC

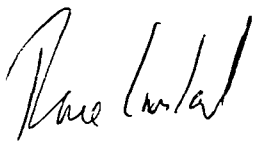
DECLARATION

The accompanying dissertation entitled "Risk in the Development of Design" is based on work carried out by the author at the University of Bristol between January 1994 and August 1997.

All of the work and ideas in this dissertation are original unless otherwise acknowledged, either in the text, or by reference.

This work has not been submitted for a degree or diploma at this or any other University.

The views expressed in this dissertation are those of the author and not those of the University.

Signed 

Date 31/8/97

CONTENTS

LIST OF APPENDICES

CHAPTER 1

Introduction

Design without risk is simply duplication - innovation implies uncertainty. There has always been risk in the development of design. Over the past few decades, driven by increasingly complex engineering designs and the growth of mass production, there has been a focus on risk assessment - many reliability techniques were developed in response to the needs of the defence industry in the Second World War and more recently work in risk assessment with regard to human harm has partly been motivated by public enquiries, such as Windscale, and investigations following accidents such as Piper Alpha. In the past ten years, with the advent of concurrent engineering and the growth of huge multi-national projects, with large, often geographically disparate design teams, the focus has broadened to cover all stages of the control and management of risk - not only technical risk, and risk of human harm, but also cost risk and time-scale risk. This first chapter introduces a research project investigating tools for the modelling of uncertainty and the assessment and management of risk - particularly cost risk - during the early stages of the design process. In the first section of this chapter the background and motivation for the research is outlined. The second section summarises the aims of the research and describes the steps taken to achieve these aims. In the third section, the structure of the thesis is presented.

1.1 Background and Motivation

The most important design decisions are made early in the design process:

> *"... for a typical product, 75 percent of the manufacturing cost is committed by the end of the conceptual phase of the design process"*. [Ullman 1992]

...and at this early stage, by definition, there will be much uncertainty present. Not only is much information regarding the designed artefact either missing or uncertain, but the problem itself is likely to be ill-defined ([Cross 1994] for example). Goals may be only vaguely expressed with many criteria and constraints still unknown. This combination of much uncertainty regarding highly significant decisions means that the presence of risk in the development of design is unavoidable.

1.1.1 Complexity and The Design Team

The number of designers involved in a typical design project has increased dramatically over the past century. One argument is that this is due to an increase in the complexity of engineering design - Ullman shows that there has been an exponential increase in complexity as the number of components in designed artefacts has risen from about 250 parts for a bicycle in 1880, through 25,000 components for an automotive

design in 1960 to more than 5 million components for the Boeing 747 aircraft in 1980 [Ullman 1992]. However, it could be argued, for example that large passenger liners were of high complexity far earlier than the Boeing 747, and that another reason why we now see larger design teams is that the design is now more completely defined by designers - in Victorian times a far greater proportion of what we would now consider to be design work was left to artisans.

Both of these factors have given rise to large design teams, who have needed to find ways to work together co-operatively (a typical automotive design team at Rover Group Ltd comprises 200 design engineers even before a decision to manufacture the new design has been taken). The relatively new research field of Computer-Supported Co-operative Work (CSCW) addresses this need and studies how it is met using computer-based systems. Technologies such as video-conferencing and the Internet, and software using these technologies (generally termed groupware) such as virtual reality systems and document management systems, all help teams to co-operate, even when geographically disparate. Large multi-national projects making extensive use of CSCW tools and techniques are not uncommon.

The spread of concurrent engineering (CE) [Sohlenius 1992] in the 1990s has lead to the involvement of manufacturing engineers, marketing experts etc. in the design process, and to a truly multi-disciplinary design team. One effect of CE on the design process is that design information is communicated at an earlier stage, when it is more uncertain, and this can mean that the impact of early design decisions is increased even further; Ward and co-workers [Ward 1995] suggest an approach termed Set-Based Concurrent Engineering (adopted by Toyota) whereby sets of alternative designs are developed in parallel for as long as possible and early design decisions are avoided. Uncertainty regarding values of design attributes is explicitly modelled, for example using interval values to represent dimensions in specifications sent to suppliers. This approach provides an example of how a shared design model incorporating uncertainty could be developed.

An approach to managing complexity which is common in engineering design is hierarchical decomposition. In [Koopman 1995] the author suggests a taxonomy of structure/behaviour/goal-based hierarchies for design decomposition, and notes that the choice of decomposition strategy depends upon both technical, organisational and business considerations. In a *structural* hierarchy, the system is decomposed into its physical components (i.e. according to its "form"), or other logical objects - the elements of a structural decomposition of a software design, for example, might include data structures and variables. In a *behavioural* hierarchy the elements of the decomposition are the "functions" fulfilled by structures, or actions or processes performed by structures or executed upon them - for example execution of a routine in a software design, or supporting a load in a mechanical design. The elements of a *goal* decomposition are the emergent design properties which satisfy the needs the design is intended to fulfil - for example performance targets, costs and aesthetic goals. All three types of hierarchy are used in different circumstances, and a design decomposition strategy may combine two or more of these types of hierarchy.

Increasingly complex artefacts and larger design teams generally lead to higher levels of uncertainty and risk - in a way which is somewhat analogous to the low reliability demonstrated by complex systems. The need for tools and techniques to help control and manage the complexity of today's large projects has led to a strong interest in project risk management in recent years. Methodologies such as Riskman [Carter 1994], the

risk engineering approach in [Cooper 1987] and the soon-to-be-published PRAM guide from the Risk SIG of the Association for Project Management, prescribe methods for the identification, recording, control and management of risks throughout the life of the project, usually along with some form of probabilistic model for risk assessment. The emphasis in modern project risk management is less on the modelling aspects, which are often very much simplified, than the recording, control and monitoring aspects. Uncertainty modelling in design is however an active research area in design theory, where both probabilistic and fuzzy approaches have been explored.

1.1.2 Uncertainty Modelling in Early Design

The purpose of performing uncertainty modelling during early design is to help improve the quality of design decisions - decisions which, by the very nature of the design process, must be made under conditions of uncertainty. The essential problem of choosing between design alternatives on the basis of multiple, possibly uncertain, attribute values has been tackled by many researchers - often motivated by the need for an evaluation function for use in optimisation algorithms in automated design tools. Research in the area of uncertainty modelling in early design is reviewed later in this thesis (Chapters 3 and 5 and Appendix B), but some significant recent work is briefly summarised here.

Wood and Antonsson's Method of Imprecision [Wood 1989] helps designers to choose input values for design attributes which satisfy a requirement on the output parameter. The method allows the designer to represent his or her level of preference for each possible value of a design attribute - this is interpreted as the membership function of a fuzzy number. The membership function of the output parameter is calculated from the membership functions for multiple input attributes using a fuzzy calculus. Knowledge of this membership function informs the designer of the degree of membership achieved by the required value for the output. It is a useful feature of the calculus employed that one can determine directly a set of values for the input parameters which would result in the required value for the output attaining a membership of 1. In more recent work [Scott 1995] the method has been applied to trade-off decisions between design alternatives - where the output parameter is a utility measure given by an aggregation function of the design attributes. The choice of aggregation function is determined by the trade-off strategy required.

Thurston and co-workers have proposed an approach to choosing between design alternatives with multiple uncertain attribute values, which is based on multi-attribute utility theory and certainty-equivalence questions. This approach allows the designer's attitude to risk to be represented (as a utility function) in an automated design optimisation tool or an expert system to provide advice to the designer [Thurston 1994]. Herling, working with Ullman, has proposed a decision support system for teams of designers based on team members assessing alternatives against multiple criteria, but where uncertainty may be incorporated into an assessment by including a measure of the designer's level of relevant knowledge [Herling 1995]. Both Thurston and Herling's models are probabilistic.

Janet Allen, at the Georgia Institute of Technology, has used fuzzy numbers to represent the constraints and goals in a design optimisation problem [Allen 1992]. In this work, the problem of complexity is addressed, as often in engineering design, via hierarchical decomposition. Peplinksi, working with Allen, has developed the FLAME software tool [Peplinski 1996], which uses an interval representation for the assessment of each

design alternative against each criterion and then uses a merit function to arrive at an interval value for the rank of each design alternative. The designer may then make the final judgement which takes into account the degree of uncertainty concerning the rank.

1.1.3 Early Design Decisions

The decision problems faced in the early stages of the design process are diverse and complex: As mentioned above, design engineers face choices between alternative designs, where each alternative may be evaluated according to several criteria, and the results of these evaluations may be uncertain. Managers must decide whether or not a design is feasible, in terms of overall cost, performance parameters or design time - none of which can be known exactly at the time when the feasibility decision is taken.

A feasibility decision may be regarded as the outcome of a bidding process which is internal to the design company. Faced with uncertain and incomplete information, the project manager must choose a price (in units of time or financial cost) or a value for design attributes such as weight or performance parameters, to include in a "bid" for the design project - this price will become the project budget if the "bid" is successful and the project is deemed feasible. In other companies, a decision-maker must choose the price for an external bid. Whether the bid is internal or external, the decision-maker should choose a price which reflects the level of risk he or she is willing to accept - including both the risk of failing to deliver for the agreed price (or attribute value) and the risk of the bid failing. This requires that early estimates for costs and other design attributes can be obtained which include a measure of their degree of uncertainty.

When approaching a feasibility decision, or the assignment of a bid price, the project manager must also decide how best to utilise existing resources (be they temporal, human or financial) so as to minimise the level of uncertainty which will be present when the decision time is reached. During this period, as development of the design proceeds, the manager requires an overview of the corresponding development of the levels of risk associated with cost and other design attributes in the project as a whole.

1.2 Aims of the Research

This thesis describes the work carried out by the author to investigate tools to support teams of design engineers and their managers in decision-making under uncertainty during the early part of the design process; at a time when much uncertainty, and hence much risk, is present.

The main aims of the work presented here were as follows:

- To gain an understanding of:
    1. The nature of the design process.
    2. The types of uncertain information which are important during the early stages of the design process.
    3. The available mathematical and computational representations for uncertainty.
    4. The process of risk management during design projects.

- To use this understanding to develop new tools to enable a team of design engineers in an arbitrary design domain to build a shared model of the designed artefact which includes uncertainty, and which may be incrementally refined to reflect the development of the design throughout the design process - termed a risk model.
- To provide methods of evaluating the risk model which:
    1. Enable design engineers and their managers to explicitly take risk into account when making early design decisions - particularly those concerned with the setting of internal or external bid prices, and the choice between discrete design alternatives.
    2. Support the project manager in monitoring the overall level of risk in the project during the development of the design and in choosing where best to concentrate resources to minimise the overall level of uncertainty.
- To test the tools with collaborating industrial partners - in particular, to test whether the tools and the underlying structure of the risk model have sufficient generality to be applicable in diverse design domains.

The steps taken to achieve these aims were:

1. A review was conducted of the design process models found in the literature and a taxonomy of types of uncertainty present in early design was developed, informed by recent research in the field.
2. An investigation was conducted into current project risk management practice - both as advocated in the literature and as actually practised by two of the collaborating industrial partners (Rover Group Ltd and Cegelec Projects Ltd).
3. The uncertainty modelling techniques currently in use in engineering in general, and engineering design in particular, were reviewed.
4. The requirements for the risk modelling tools were identified, informed by the findings from 1., 2. and 3.
5. A risk modelling methodology was developed. The underlying structure for the risk model was defined, together with the process by which it may be refined during the design process.
6. A software tool was written to implement the methodology - i.e. to enable design engineers and their managers to build and evaluate risk models.
7. A technique termed hierarchical risk sensitivity analysis was developed for identifying the contribution to overall uncertainty made by each part of the risk model.
8. The methodology and software tool were evaluated using case studies conducted with Cegelec Projects Ltd and Rover Group Ltd.

The collaborating companies were Cegelec Projects Ltd, Rover Group Ltd and Ove Arup and Partners. Cegelec Projects design large scale electrical installations, such as ship-board control systems, steel-rolling mills and other types of industrial processing systems. They design and supply to their customers complete systems including all the hardware (sensors, controllers, workstations etc.), software and cabling required for

the application. The two simplified case studies considered in this research project concerned the design of a shipboard positioning system and of a steel roughing mill. Rover Group Ltd are an automotive design and manufacturing company. During this research, the case studies have concentrated on the interior trim area for a new vehicle program, particularly the facia. The facia is the area in the front of the interior of the vehicle, comprising the instrument panel, binnacle, gear-lever, glove box etc. The risk model developed for Rover was the largest and most detailed of the case studies. Ove Arup work in civil engineering design and and also in the area of vehicle crash-simulations. Although these diverse design domains informed the research indirectly, no case studies were completed with Ove Arup.

1.3 Structure of the Thesis

The thesis is divided into twelve chapters, with this Chapter (the Introduction) as the first. The contents of the remaining eleven chapters are as follows:

Chapter 2: Risk

The meaning of the term is explored, and the definition adopted in this thesis is given. Theories of perception of risk are briefly reviewed.

Chapter 3: The Problem Domain: Design

Models of the design process taken from the literature are reviewed, and this is followed by a discussion of the types of uncertainty which are present in early design. Recent research in the area of uncertainty modelling in early design is summarised. A classification of uncertain design information is presented and this leads to a view of the types of information which the risk model should be able to represent.

Chapter 4: The Problem Domain : Management

The nature of the design project risk management process is explored; a review of project risk management methodologies taken from the literature is followed by investigations into current practice in risk management at Cegelec Projects Ltd and Rover Group Ltd. The requirements for the risk tool are summarised - the full requirements document, annotated to indicate whether requirements were met, transformed, abandoned or not implemented, is reproduced in **Appendix A**. One of the key requirements which is identified here is an object-based structure for the risk model.

Chapter 5: The Representation of Uncertainty

The representations of uncertainty used in engineering design are summarised, including intervals and support pairs, probability and fuzzy representations. A detailed review of the techniques used to propagate uncertainty in each case and the engineering application areas is given in **Appendix B**. The representation chosen for the risk model is presented and justified.

Chapter 6: Object-Orientation

The object-oriented (OO) paradigm is introduced and summarised and the reasons for choosing this paradigm for the risk model are presented. An OO analysis, design and implementation method called Fusion is described, as is the OO programming language C++ which was chosen for development of the software

tool. The Fusion method is used both as a modelling notation for the risk model and in the development of the software tool, and its choice for these purposes is justified here.

Chapter 7: Case Studies of Present Design Practice

The case studies which are eventually used to evaluate the methodology and software tool are first introduced in this chapter, and described with the aid of Fusion graphical views. The design of a steel roughing mill and a shipboard positioning system at Cegelec Projects Ltd are described in a simplified form, and the design of the interior trim in a new vehicle development at Rover Group Ltd is developed in more detail. Variant designs are differing versions of a component which will be selected according to the exact vehicle model chosen by the customer - it is explained in this chapter that, even during the early design stage, such variant designs are important.

Chapter 8: The Design Modelling and Risk Assessment Methodology

The new OO risk modelling methodology which has been developed is described in this chapter.

Chapter 9: Methods of Evaluating Risk

The mathematical and computational techniques used by the software tool are described. In particular, the sampling techniques which are used in an evaluation of the risk model are detailed together with the hierarchical risk sensitivity analysis technique which has been developed and also a technique which has been developed for modelling different configurations of variants (configuration modelling) under uncertainty.

Chapter 10: The Risk Tool (RiTo)

The implementation of the software tool is described - its structure and its interaction with a commercial Fusion CASE tool and with data files is explained. The capabilities of the software tool are described from the user's perspective. Some comments on the design of the software tool and on the use of the Fusion CASE tool used are presented. A more detailed description of the design of the software tool is given in **Appendix K**, with the aid of Fusion graphical diagrams and schemata.

Chapter 11: Implementing Models of the Case Studies

The case studies introduced in Chapter 7 are re-visited. They are modelled using RiTo and the risk models themselves are presented, as are the evaluation results. The results obtained for the steel roughing mill case study are compared with those obtained using a model constructed with a commercially available risk software package. The hierarchical risk sensitivity analysis algorithm (presented in Chapter 9) is explored using the interior trim case study and where practical the results are verified using analytical methods or more direct numerical methods.

Chapter 12: Conclusions and Further Work

The findings from the research are summarised, and directions for further research building on the ideas expressed in this thesis are proposed.

1.4 References

[Allen 1992] Allen, J. K., Krishnamachari, R. S., Masetta, J., Pearce, D., Rigby, D., and Mistree, F., "Fuzzy compromise: an effective way to solve hierarchical design problems", *Structural Optimization*, vol. 4, pp. 115-120, 1992.

[Carter 1994] Carter, B., Hancock, T., Morin, J-M., and Robins, M., "*Introducing RiskMan Methodology: The European Project Risk Management Methodology*", Oxford, UK: NCC Blackwell Ltd, 1994.

[Cooper 1987] Cooper, D. F. and Chapman, C., "*Risk Analysis for Large Projects: Models, Methods and Cases*", Chichester : John Wiley and Sons, ISBN 0 471 91247 6, 1987.

[Cross 1994] Cross, N., "*Engineering Design Methods: Strategies for Product Design*", Second Edition, Chichester, UK: John Wiley and Sons, ISBN 0 471 94228 6, 1994.

[Herling 1995] Herling, D. and Ullman, D. G., "Engineering Decision Support System (EDSS)", *Proceedings of the 7th International Conference on Design Theory and Methodology*, ed Ward, A. C., held as part of the ASME Design Engineering Technical Conferences, Boston, Mass., DE-Vol 83, Vol. 2, pp. 619-626, September 1995.

[Koopman 1995] Koopman, P. J., "A Taxonomy of Decomposition Strategies Based on Structures, Behaviours and Goals", *Proc. Design Engineering Technical Conferences*, ASME, Boston,, vol. 2, pp. 611-618, 1995.

[Peplinski 1996] Peplinski, J. D., Koch, P. N., Allen, J. K., and Mistree, F., "Design Using Available Assets: A Paradigm Shift in Design for Manufacture", *Concurrent Engineering: Research and Applications*, vol. 4, no. 4, pp. 317-332, December 1996.

[Scott 1995] Scott, M. J. and Antonsson, E. K., "Aggregation Functions for Engineering Design Trade-Offs", *Proceedings of the 7th International Conference on Design Theory and Methodology*, ed Ward, A. C., held as part of the ASME Design Engineering Technical Conferences, Boston, Mass., DE-Vol 83, Vol. 2, pp. 389 - 396, September 1995.

[Sohlenius 1992] Sohlenius, G., "Concurrent Engineering", *Annals of the CIRP*, vol. 41 issue 2, pp. 645-655, 1992.

[Thurston 1994] Thurston, D. L. and Crawford, C. A., "A Method for Integrating End-User Preferences for Design Evaluation in Rule-Based Systems", *Transactions of ASME- Journal of Mechanical Design*, vol. 116, pp. 522-530, June 1994.

[Ullman 1992] Ullman, D. G., "*The Mechanical Design Process*", New York : McGraw-Hill ISBN 0-07-112871-9, 1992.

[Ward 1995] Ward, A., Liker, J. K., Cristiano, J. J. and Sobek, D. K. II, "The Second Toyota Paradox: How Delaying Decisions Can Make Better Cars Faster", *Sloan Management Review*, pp 43 - 61, Spring 1995.

[Wood 1989] Wood, K. L. and Antonsson, E. K., "Computations with Imprecise Parameters in Engineering Design: Background and Theory", *Transactions of the ASME*, vol. 111, pp. 616 - 625, December 1989.

CHAPTER 2

Risk

In this chapter the meaning of risk is explored. The first section contains a number of different definitions of the term as used in various fields, and the definition adopted in this thesis is stated. In the second section a brief overview is given of current theories concerning the ways in which people perceive risk.

2.1 What Does "Risk" Mean?

The term "risk" is used to signify several subtly different concepts, in areas as diverse as health and safety, insurance, economics, engineering design and project management. What is clear is that uncertainty may lead to risk, and that this relationship between risk and uncertainty depends upon the perceived significance of the consequences of the uncertain event. Thus we cannot consider risk without considering uncertainty, although uncertainty may be present without incurring risk. Some definitions of risk used in the literature are:

> *Hazard, chance of bad consequences, loss, etc., exposure to mischance.*
> [Shorter Oxford English Dictionary]

> *Risk is the possibility of suffering loss, injury, disadvantage, or destruction.*
> [Webster's Third New International Dictionary 1981]

> *Risk is exposure to the possibility of economic or financial loss or gain, physical damage or injury, or delay, as a consequence of the uncertainty associated with pursuing a particular course of action.*
> [Cooper 1987]

> <Project risks are defined as> *undesirable events, such as problems and accidents, which cause project delays, cost overruns, or deficiencies in technical performance.*
> [Niwa 1989]

> *A risk involves uncertainty and has an impact.*
> [Carter 1994]

> <Risk is the> *perception of the probability and magnitude of some future adverse event.*
> [Adams 1995]

> *Risk is the potential for realization of unwanted negative consequences of an event.*
> [Rowe 1977]

> *Risk is the measure of the probability and severity of adverse effects.*
> [Lowrance, William W., Of Acceptable Risk 1976]

> *What distinguishes an acceptable risk problem from other decision problems is that at least one alternative option includes a threat to life or health among its consequences. We shall define risk as the existence of such threats.*
> [Fischhoff 1981]

In economic and decision theory, decision making "under risk" is interpreted as implying a complete knowledge of the underlying probability distribution, whereas "under incomplete knowledge" implies only a partial knowledge of the distribution and "under uncertainty" implies no knowledge whatsoever. This definition is too formal and constraining for use in this thesis, because it effectively pre-supposes a frequentist view of probability theory and makes little sense in the personalist view. In the frequentist view, the probability of an event is a measure only of how frequently it has happened in the past - whereas the personalist view allows probability to be used as a measure of "degree of belief" (for example that an event will occur in the future). In this thesis the personalist view is adopted - see Chapter 5.2 "The Choice of Representations".

In the insurance field, the concept of risk is sometimes related to the difference between actual and expected results, or the probability of such a difference (see [Niwa 1989, p. 15]). This definition differs from those quoted above because it is not objective - it depends not only upon the objective situation but also upon what the observer expects to happen. Therefore this definition is not useful when discussing a single risk model which may be edited and observed by many people, each with their own expectations. In order to be able to communicate risks, we must have a more objective definition. In [Eeckhoudt 1995], a book concerning the economic theory of risk, largely based on utility theory, the author proposes several equivalent definitions for an increase in (numerically measured) level of risk. These differ from the traditional measure of level of risk, which is variance. His definition, most simply stated, is that an uncertain situation A has a higher level of risk than an uncertain situation B, if, and only if, all risk averse people would prefer B to A. Given the definition of risk aversion used in utility theory, this amounts to a risk measure which is a combination of the likelihood, impact and desirability (in economic theory this is simply whether it is a profit or a loss, not a subjective measure) of an uncertain event. Note however, that Eeckhoudt's measure of the quantity of risk is quite separate from the level of risk aversion of the decision maker (measured using a utility function); it does not depend upon the observer.

In general, the term is used to signify both an undesirable and uncertain event and also to signify the combined probability and impact of that event. In this thesis, the term is used in both these senses:

> *A risk* is defined to be any uncertain event which may have a detrimental impact on the outcome of the design project. The impact may be measured in units of elapsed time, financial cost or quality

(for example performance parameters in engineering design). Conversely, an *opportunity* is defined to be any uncertain event which may have a desirable impact on the outcome of the design project.

The level of *riskiness* (also often simply termed *risk*) is defined to be a combination of the probability of occurrence of an uncertain and undesirable event and the impact of that event. And the level of *opportunity* is defined to be a combination of the probability and impact of a desirable uncertain event.

A *risk model* represents both risks and opportunities and can be used to evaluate both level of riskiness and level of opportunity.

An event may be uncertain because a random or pseudo-random process is involved (e.g. the failure of an electrical component, or the occurrence of heavy rain-fall on a particular day) or it may be uncertain from the point of view of an observer simply because they have insufficient knowledge to determine whether or not the event will occur (e.g. the failure of a product launch because a competitor launches a very similar product a little earlier).

Having chosen a definition for the term, we can now consider the ways in which human perceptions of risk sometimes differ from "objective" measures such as expected values.

2.2 Theories of Risk Perception

It is often difficult to divorce the objective level of risk present from that which is perceived. The area of risk analysis and management in public policy is concerned not only with the narrow definitions of risk used by experts, but must also, by necessity concern itself with the broader view of risk taken by the general public. Major decisions concerning transport policy, environmental issues and public health cannot be made purely, for example, on a statistical basis - there are fundamental value judgements bound up with these decisions and in a democracy the values of the general public must be taken into account. Also, people's behaviour and personal decisions are strongly influenced by their perceptions of risk - decisions with public implications such as how fast to drive their car, whether or not to smoke, etc. For these reasons research has been conducted by experimental psychologists and social scientists into perceptions of risk and there have been several important and generally applicable findings (see [Morgan 1993] and [Parkin 1996, pp. 90-96]).

Slovic and co-workers [Slovic 1992] found that there are three sets of factors, which tend to occur together, which have a strong influence on the perceived level of risk. The first is the "dreadfulness" of the risk event - very high impact, catastrophic events (even if they have very low probability) which involve innocent people, are dreaded disproportionately. The second is concerned with how well the risk is understood - unfamiliar and unknown risks, with possible effects far into the future, are perceived as having a very high risk level - and the third set of factors concern the number of people exposed.

Stallen and Thomas found that people perceive situations in which they personally have a high degree of control as being less risky than those where a third party, expert or government has control. Tversky and Kahneman [Tversky 1982] discovered the "heuristic of availability" - people tend to assume intuitively that

the ease with which they can recall or imagine examples of an event is related to the likelihood of that event. Thus, people generally underestimate the frequency of common but rarely-reported causes of death such as strokes, and overestimate the frequency of news-worthy causes of death such as botulism poisoning.

The "amplification" effect whereby small incidents can give rise to large public reaction in certain circumstances is identified and modelled in [Kasperson 1988]. In [Parkin 1996, pp. 94-95], the idea that minor risks can be "stigmatised" by an association, however weak, with "dreaded" technologies is presented, with examples taken from the work of Slovic.

John Adams [Adams 1995] proposes a compensation model of risk, where people act so as to maintain a target level of perceived risk, as though governed by a "risk thermostat". If external measures are taken which reduce the risk level to which a person is exposed, for example legal enforcement of seat-belt wearing, Adams contends that the person will often respond by changing their actions so as to return the overall level of risk to the target, for example by driving their car at a greater speed. He claims that the accident statistics support his view and denies that there is generally a decrease in accident road death tolls when countries introduce seat-belt legislation - for example attributing the reduction in UK road deaths following legislation jointly to an anti- drink-driving campaign in 1983 and to an existing downwards trend, and claiming that the legislation resulted in an increase in pedestrian and cyclist deaths. It must be stated however that overall the number of road deaths per vehicle on the roads in the UK has fallen by a factor of more than 8 since 1950 [DoE 1997] - the number of licensed motor vehicles having increased from 4.4 million to 26.3 million and the annual number of deaths having actually decreased from 5,012 to 3,598. To deny that this impressive fall was largely brought about by tighter legislation and technological improvements in vehicle design does not seem reasonable - the national average level of acceptable risk can hardly have changed so dramatically - so clearly the "risk thermostat" effect can in fact be overridden by external risk reduction measures.

Further, Adams proposes a model whereby the feedback mechanism for the thermostat is subject to "cultural filters" which embody the subjective values and attitudes of the risk-taker. He therefore argues that the whole concept of "objective risk" is flawed. For example, an environmentally concerned cyclist and an individualistic car driver will never agree on the risks associated with cars as a means of transport because of their different cultural filters, and Adams argues that there is no underlying "objective risk" for them to uncover in their arguments, that the primary nature of the risk they are discussing is subjective. Whilst it is clearly true that there can be more to people's perception of risk than mere statistical fact, and that the broader concept of subjective risk perception must be respected, it is also a fact that there are many decision-making situations in daily life where simple statistical facts are available and are relevant and yet many people do not understood or use them [Paulos 1988].

It is not only lay-people who have difficulty estimating likelihood values - expert probability judgements elicited from insurance underwriters for example, are also subject to systematic biases [Bolger 1994]. Bolger and Wright consider both coherence (the probabilistic consistency of the elicited information) and calibration (agreement with available frequency statistics) as measures of validity. They suggest that the usual approach of eliciting simple constituent conditional or marginal distributions and then combining them algorithmically does not necessarily overcome these problems - the results will be coherent, but the process of re-

combination may amplify any calibration errors in the constituent distributions. Performance feedback and the framing of questions in terms familiar to the expert are amongst the solutions which they propose.

Finally, when considering theories of risk perception, it must be noted that despite the biases to which human risk judgement is subject, an over-emphasis on statistical (historical) data, when such data is sparse or of dubious relevance, can lead to an under-valuing of the subjective probability judgements which may in fact be the best information source available.

2.3 References

[Adams 1995] Adams, J.,"*Risk*", London : UCL Press, ISBN 1-85728-067-9 HB, 1995.

[Bolger 1994] Bolger, F. and Wright, G., "The quality of expert probability judgement:issues and analysis", *Expert Systems*, vol. 11, no. 3, pp. 149-158, August. 1994.

[Carter 1994] Carter, B., Hancock, T., Morin, J-M. and Robins, M., "*Introducing RiskMan Methodology: The European Project Risk Management Methodology*". Oxford, UK: NCC Blackwell Ltd, 1994.

[Cooper 1987] Cooper, D. F. and Chapman, C., "*Risk Analysis for Large Projects: Models, Methods and Cases*", Chichester, UK : John Wiley and Sons, ISBN 0 471 91247 6, 1987.

[DoE 1997] The Department of the Environment, Transport and the Regions, "*Road Accident Great Britain: 1996, The Casualty Report*", UK: Government Statistical Service, 1997.

[Eeckhoudt 1995] Eeckhoudt, L. and Gollier, C., "*Risk evaluation, management and sharing*", New York : Harvester Wheatsheaf, ISBN 0-7450-1592-1, 1995.

[Fischhoff 1981] Fischhoff, B., Lichtenstein, S., Slovic, P., Derby, S. and Keeney, R., "*Acceptable Risk*", Cambridge: Cambridge University Press, ISBN 0-521-24164 2, 1981

[Kasperson 1988] Kasperson, R. E., Renn, O., Slovic, P., Brown, H., Emel, J., Goble, R., Kasperson, J. and Ratick, S., "The social amplification of risk: a conceptual framework", *Risk Analysis*, vol. 8, no. 2, 1988.

[Morgan 1993] M. Granger Morgan, "Risk Analysis and Management", *Scientific American*, pp. 24-30, July 1993.

[Niwa 1989] Niwa, K., "*Knowledge-Based Risk Management in Engineering: A Case Study in Human-Computer Cooperative Systems*", New York : John Wiley and Sons, ISBN 0-471-62893-X, 1989.

[Parkin 1996] Parkin, J., "*Management Decisions for Engineers*", Heron Quay, London : Thomas Telford Publishing, ISBN 0 7277 2501 7, 1996.

[Paulos 1988] Paulos, J. A., "*Innumeracy: Mathematical Illiteracy and Its Consequences*", London : Viking, ISBN 0-670-83008-9, 1988.

[Rowe 1977] Rowe, W. D., "*An Anatomy of Risk*", New York : John Wiley & Sons, 1977.

[Slovic 1992] Slovic, P., "Perceptions of risk: reflections on the psychometric paradigm", *Social Theories of Risk*, ed. Krimsky, S. and Golding, D., Westport, USA : Praegar, 1992.

[Tversky 1982] Tversky, A. and Kahneman, D., "Judgement under uncertainty: heuristics and biases", *Judgement under Uncertainty: Heuristics and Biases,* ed. Kahneman, D. et al, Cambridge, England : Cambridge University Press, 1982.

CHAPTER 3

The Problem Domain: Design

Chapters 3 and 4 present the evolution of the requirements for a model which may be used to assess risk in the early stages of the development of a design. In this chapter the types of uncertain information which such models need to be able to represent are determined. In the first section, the nature of the design process is explored in a review of design process models and general conclusions are drawn regarding the nature of the design process which the risk model should support. The second section contains a brief review of current research into uncertainty modelling in early design to support design decision-making, and an abstract discussion of the occurrence of uncertainty in early design. A classification of uncertain design information is proposed. This classification is used to clarify the types of uncertain information which the model should be able to represent.

3.1 Theories and Models of the Design Process

3.1.1 Overview

Design is an activity which is characterised by decision-making under uncertainty, from specification and market requirements through to judgements of performance in practice. The presence of uncertainty is a defining feature of design and thus any design model - physical, cognitive or computational - will be subject to uncertainty during the design process. Antonsson and Otto state:

> *...imprecision is an integral part of the engineering design process. Not imprecision in thought or logic, but rather the intrinsic vagueness of a preliminary, incomplete design description.*
> [Antonsson 1995]

The degree or magnitude of uncertainty to which a design model is subject is generally reduced during the design process (see Figure 3-1) as a result of design activities.

Further, Ullman, Ringstad and others agree that the rate of acquisition of knowledge about the design problem generally decreases with time; thus, there is disproportionately more uncertainty present towards the beginning of the design process than towards its end. Later in this chapter, in Section 3.2.6, a categorisation of uncertainty in early design will be presented, addressing the question:

"How can we classify the types of uncertain information which are important during the early phases of the design process?"

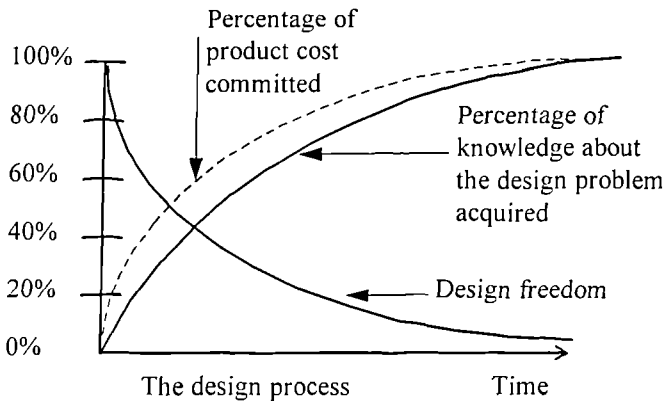


Figure 3-1: Increase in knowledge (and consequent reduction in uncertainty) during the design process, from [Ullman 1992] and [Ringstad 1996]

However, an understanding of the nature of the design problem and process is a necessary prerequisite to defining such a categorisation. To a large extent, the nature of the design process itself determines the nature of the uncertain design knowledge which exists during the process. Two quite different design processes, for example pursued by a human designer and an automated design tool, could conceivably result in very similar certain properties for the two designed artefacts at the end of the design process. However, the nature of the uncertain knowledge acquired during the early stages will be very different for the human designer and for the automated design tool. Thus, in this section, a short review of theories and models of the design process is presented.

All design problems have a goal, some constraints within which the goal must be achieved and some criteria by which the goodness of a design solution can be evaluated. However, writers on the theory of design agree that design problems are generally **ill-defined**. In the textbook "Engineering Design Methods: Strategies for Product Design" [Cross 1994], the author expands on what is meant by "ill-defined" in this context:

1. There is no definitive formulation of the problem.
   Goals are usually vague. Many constraints and criteria are usually unknown. The problem context may be poorly understood. Formulations of the problem may be fixed, but then may change later in the design process.

2. Any formulation of the problem may embody inconsistencies.

3. Formulations of the problem are solution-dependent.
   It is difficult to state the problem without reference to a solution concept. The designer's perception of solution concept/s will influence how they state the problem.

4. Proposing solutions is a way of understanding the problem.

5. There is no definitive solution to the problem.

Early models and theories of the process of solving such ill-defined design problems broadly fall into two categories. Firstly, there are those which have been developed by formal observation, recording and subsequent analysis of designers at work (known as protocol studies) - examples include the models of Ullman *et al.*, and Esterline *et al.* (described later in this section). Secondly, there are those which have arisen from attempts to define a prescriptive model of the design process (a "canonical design process"), which are not necessarily based on formal observation but on the practical design experience of the authors. Such models may appear less well-founded but they make intuitive sense to many designers and are widely recognised. The most well known examples are the models of Pahl & Beitz, the German VDI, Hubka & Eder and of Pugh. These will also be described in more detail later.

The two approaches have some common features. In all of these models or theories, the design process is divided into several distinct design phases. The phases involve problem representation followed by solution generation and then a search for the "best" solution - where the search consists of an evaluation of solutions and choice.

More recently, models such as those of Cross, Bond & Ricci and McMahon *et al.*, for example (all described later), have been built on the joint basis of the distilled experience expressed in the prescriptive models and the empirical results expressed in the descriptive models. These models move away from the paradigm of serial phases (possibly iterated) and towards concurrency and parallelism.

One of the major culture changes in engineering design over recent years has been the introduction of concurrent engineering (CE) [Sohlenius 1992]; the phrase was coined in the USA in 1989. Prior to the introduction of CE the process of transforming a concept into a product (the product design phase) traditionally involved completion of the design description before preparation for manufacturing commenced, at which point the design was "thrown over the wall" from the design department to the manufacturing department (see Figure 3-4 "Phases in the VDI 2221 model of the design process", for example). Concurrent engineering, however, aims to shorten development times by the simultaneous evolution of the product and the manufacturing and design processes. Designers, manufacturing engineers, marketing experts and so on comprise a dedicated, multi-functional team (often co-located), collaborating during the design process.

Because of the difficulty of co-ordinating such a team, the approach tends to require a highly structured development process, with detailed manuals of development process plans. Intensive communication between team-members is required and decisions must often be made earlier than with traditional serial development - such early decisions are frequently based on incomplete or uncertain data.

The aim of CE is to "zig-zag" through the domains illustrated in Figure 3-2, visiting them all at each stage in the design process, developing them all concurrently. Probably the most important predecessor (and contributing method) to CE is "design for manufacturing and assembly", developed by Boothroyd and Dewhurst in the early 1980s [Boothroyd 1987] [Boothroyd 1994]. DFMA is a design methodology, supported by computer tools, which aims to reduce the assembly and manufacturing costs of a design so far as possible whilst still providing the required functions. Systems have been developed which assign a rating

to the "manufacturability" of a design, and suggest methods for improving it. Improvements are made, for example, by taking advantage of economies of scale (e.g. designing parts which can be used in multiple products and by reducing the total number of part types in a product), by using standardised parts so far as possible and by choosing simple, low-cost operations over expensive, new, technologies. Design choices should be made which facilitate the assembly process - for example aiming for a simple, modular construction and choosing fast and efficient insertion, fastening and joining methods.

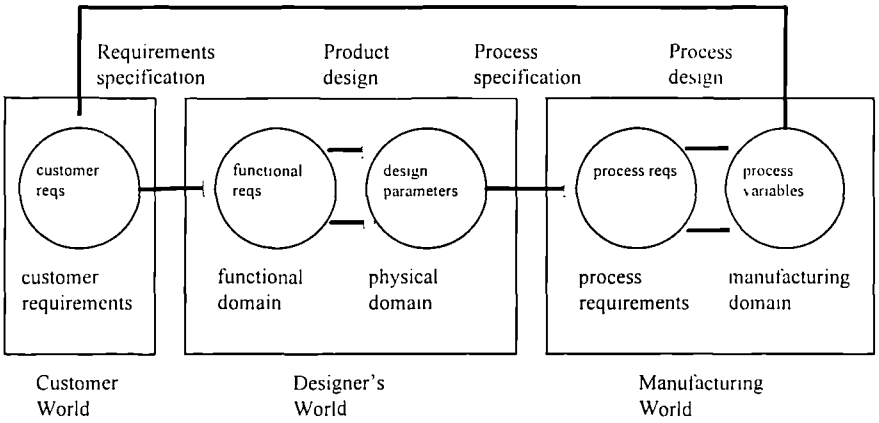


Figure 3-2: The Domains of Engineering Product Development, from [Sohlenius 1995]

Quality Function Deployment (QFD) is another important tool; this is a methodology, introduced in Japan in the 1970s, which involves recording and quantifying the customer requirements and translating them into measurable engineering targets as an aid to understanding the design problem and to evaluating proposed product designs. Other tools important to CE include Failure Mode and Effect Analysis (FMEA) and Taguchi's method for robust design (both described in Appendix B). The set-based CE model proposed by Ward *et al.* (described later in this section), which involves the simultaneous development of sets of design solutions, is particularly interesting from the perspective of uncertainty in design.

A second defining characteristic of the models discussed below, in addition to the degree of their concurrency, is the degree of evolutionary behaviour which they embody. As noted in [Konda 1992], the morphological methods prescribed by the canonical school (which aim to generate as many alternative design solutions as possible) attempt to model engineering design problems as though they were problems in natural science; but many writers [Hillier 1972] [Darke 1984] insist that design must rely on prior knowledge of solution types and it can thus be regarded as an evolutionary process. A brief description of each model follows:

3.1.2 Pahl & Beitz

In [Pahl 1988, p. 4] Pahl and Beitz divide design problems into three categories, which are widely accepted:

original	- Requires an original solution principle.
adaptive	- Adapt previous design to new task (may need original components).
variant	- Solution principle and task remain the same, vary some design parameters.

The phases of the design process are illustrated in Figure 3-3.

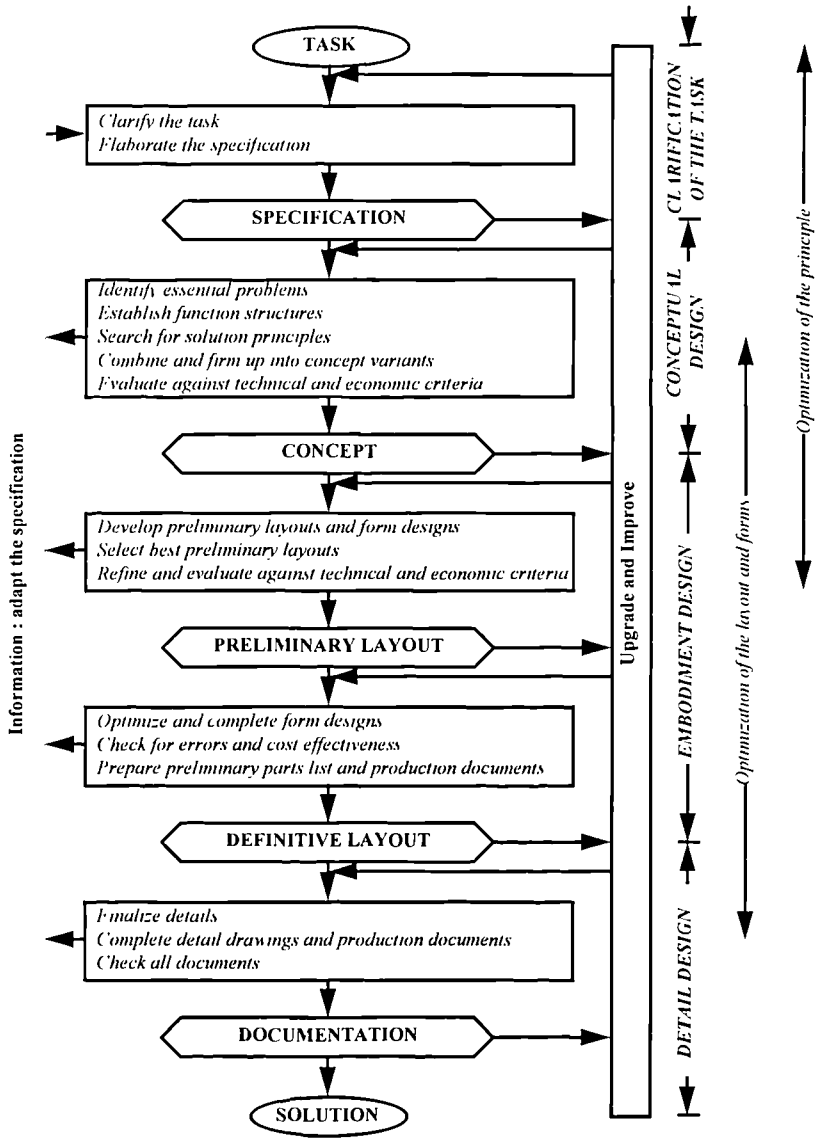


Figure 3-3: Pahl & Beitz's model of the design process, adapted from [Pahl 1988]

A brief description of each phase follows:

Clarification

Collecting information about requirements and constraints then generating a detailed requirements list.

Conceptual design

Identification of solution principles, generation of possible solutions - "concept variants". Evaluation of concept variants, resulting in a choice of one or more to be developed further in the embodiment design phase. The evaluation criteria here are both economic and technical.

Embodiment design

Generation of possible forms and layouts from concept/s. Technical and economic evaluation of layouts, resulting in the choice of a single layout to be developed further in the detail design phase.

Detail design

Determination of arrangement, form, dimensions, materials and surface properties of all components. Generation of production documents. Repetition of economic and technical evaluation.

Note that at any stage it may become necessary to adapt the specification, using new information which has been generated as a result of the design activities. As Cross notes, above, in an ill-defined design problem with no definitive problem formulation and where formulations of the problem are solution dependent, the specification will evolve during the design process.

Writers on the canonical design process broadly agree on the categories and phases identified above. In their review paper, Finger and Dixon [Finger 1989] quote another (unpublished) review paper [Juster 1985] which identifies the following areas of agreement amongst writers on the canonical design process:

Design problems can be divided into three types:

- Original or new designs.
- Transitional or adaptive designs.
- Extensional or variant designs.

The designer proceeds iteratively through the stages:

- Recognition of need.
- Specification of requirements.
- Concept formulation.
- Concept selection.
- Embodiment and detail design.
- Production, sales and maintenance.

There are three stages of thought in design:

- Divergence: synthesis, extending the design boundary.
- Transformation: bounding the problem, making judgements, decomposing the problem, modifying sub-goals.
- Convergence: analysis, progressive reduction of secondary uncertainties until a single design emerges.

It is during the most creative stage, that of divergence, that the designer's experience and intuition have the most powerful influence. How does a designer decide which of all the scientifically possible concept variants

and form/layout variants he or she will generate and evaluate? In any realistic design situation, experience and intuition clearly must limit the search space considerably so that the design process can be completed within the time and budget resources available.

3.1.3 VDI

In Germany, the professional engineers body, the Verein Deutscher Ingeniere (VDI) has produced various guidelines prescribing a design process in an attempt to standardise the design process across industries. One of these is VDI 2221 "Systematic Approach to the Design of Technical Systems and Products". The phases of the design process in the VDI model are shown in Figure 3-4. In this model, having clarified the problem in stage 1, the designer in stage 2 then determines the required functions and decomposes them into sub-functions or sub-problems, forming a "function structure". In stage 3, a solution is sought for each sub-problem and these solutions are combined to provide an overall solution principle, which is then taken through embodiment and detail design in stages 4, 5 and 6.

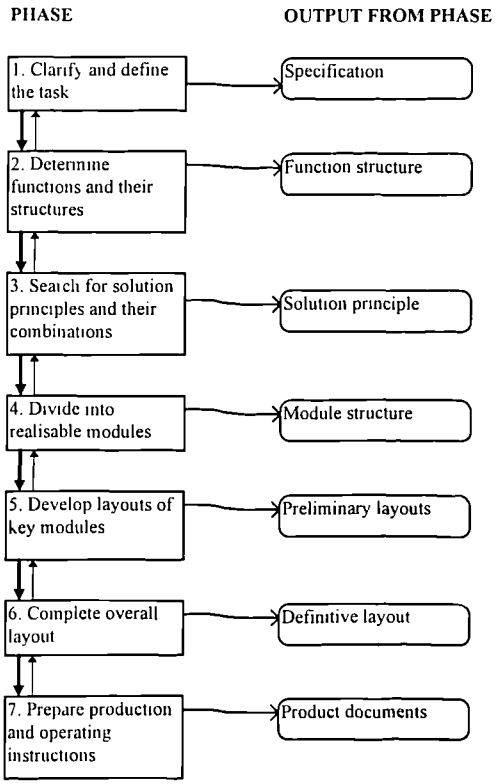


Figure 3-4: Phases in the VDI 2221 model of the design process

The traditional designers' approach to ill-defined problems is to propose one or more solution concepts fairly early on and then use them to further define the problem. The VDI process differs from this traditional approach in that it is problem-centred, rather than solution-centred; it could be argued that the designer thus loses sight of the overall solution towards which they are working, whilst solving individual sub-problems. The VDI approach also assumes that the design problem can be decomposed such that combining good solutions to the sub-problems will yield a good solution to the parent problem; but not all design problems can be decomposed in this way.

3.1.4 Hubka & Eder

Hubka and Eder [Hubka 1984] have developed a well-known theory which describes technical systems (e.g. a designed artefact), technical processes and also the design process itself in terms of *transformation processes*. Formally, a transformation process is an artificial procedure during which some properties of an *operand* are subjected to a change; during the transformation process, sequences of operations occur during which *operators* affect the operand by moving materials, energy or information.

The design process is itself modelled as an information transformation system (Figure 3-5); transforming requirements into a manufacturing description for the product. The operators (designers, the tools they use, design information and the management of the designers) all affect the operand (the design) during the design process. Operations during the design process include a sequence of one-to-many mappings from an abstract representation to several more concrete representations (design proposals); design operations which follow this refinement include evaluation, selection, improvement, quantification, optimisation and analytical operations to determine design properties. Essentially, the design process prescribed is a sequence of increasingly detailed phases of "concretization"; at each phase, several alternatives are synthesised, analysed and evaluated and a single selection is made which is then carried forward to the next phase. The aim is to intelligently minimise the number of alternatives that must be considered by pruning a search tree, thus avoiding a "combinatorial explosion". Iteration is introduced into this procedural and hence totally serial model by the inclusion of a feedback loop but "there should be as little iteration and backtracking as possible in the design process" (p 216).

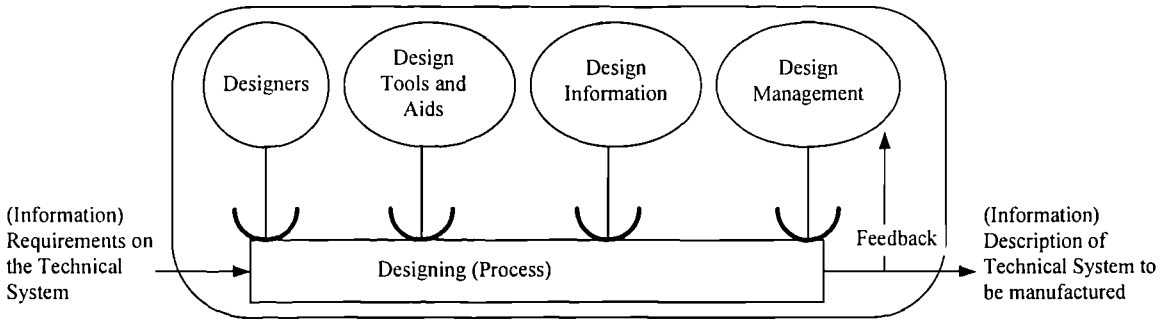


Figure 3-5: The Design Process as an Information Transformation System, from [Hubka 1984]

It is interesting to note that this model (like all the canonical models and most of the others mentioned below) accepts the precepts of Artificial Intelligence: that fundamentally, design activity, like all intelligent activity, consists of

1. Representation of the (design) problem as a pattern of symbols.
2. Performing operations on the patterns to generate a set of potential solutions.
3. Search to select a (design) solution from amongst these possibilities.

This characterisation of intelligence, the basis of the "physical symbol system hypothesis" (see [Luger 1993, p. 29 and p. 689]), is the foundation of Artificial Intelligence. Thus, the simplest thing which we can say about the design process is that, possibly at some very low level, designers proceed by generating potential solutions and then testing them. There is a view, which is implicit in the canonical design process models,

that this characterisation of the design activity can be usefully abstracted to quite a high-level. A conflicting view is that in design the search space cannot be sufficiently known or defined for any but the most trivial of problems.

3.1.5 Pugh

In Pugh's "total design activity model" [Pugh 1991], he defines a central core of activities which are necessary for any design activity - a "design core". This consists of specification, concept design, detail design, manufacture and sales (in broad agreement with the general stages of canonical design identified by Juster, above). One of Pugh's major contributions to canonical design has been his "method of controlled convergence", otherwise known as Pugh's method. This is a simple but effective method for the evaluation and hence selection of design concepts; it is mentioned here because of its impact on the prescribed design process.

In Pugh's method of controlled convergence, a 2-dimensional matrix is drawn up with the concepts under consideration along one axis and the criteria against which they are to be evaluated along the other. One concept is chosen to be the datum, against which all others will be compared (either a comparable competitive product or the initially preferred concept). The matrix is then populated - each <concept, criterion> pair is given a value of "+" (meaning "better than datum"), "-" (meaning "worse than datum") or "S" (meaning "same as datum"), and the total number of "+"s, "-"s and "S"s are then aggregated for each concept. This gives an idea of the relative strengths and weaknesses of the concepts.

Attempts are then made to improve both weak and strong concepts - and where this proves possible, the modified concepts are added into the matrix, expanding it. Those weak concepts which could not be improved are then removed from the matrix, contracting it. If new solutions arise, they too are added into the matrix. This expansion and contraction of the matrix is continued until a small number of concepts emerge as strong. These strong concepts are then developed further and in more detail before possibly repeating the whole exercise.

Figure 3-6 illustrates the resultant evolution of concepts during the design process. As Juster notes, above, the alternation illustrated between convergent and divergent thought is a hallmark of the design process.

In principle, Pugh's method prescribes a design process characterised, like that of Hubka & Eder, by the generation of potential solutions followed by test to search for the "best" solution. However, it is far more evolutionary in nature since attempts are then made to improve upon a small set of solutions, rather than relying on generation of a large number of initial solutions.

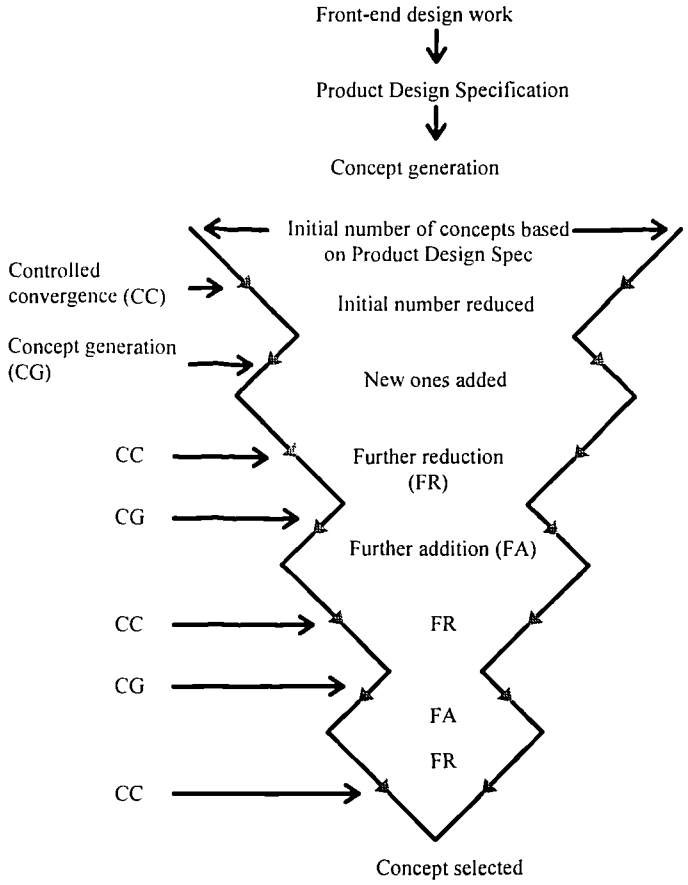


Figure 3-6: Concept Generation and Selection, from [Pugh 1991]

3.1.6 Taguchi

No discussion of the design process is complete without reference to the work of Dr Genichi Taguchi and the field of quality engineering. Taguchi originated the ideas of the "quality loss function" and of "robust design" [Taguchi 1986]. The concept behind the quality loss function is that any deviation of a characteristic of a designed artefact from its optimum value gives rise to a loss of quality in some respect. The price for this loss may be paid by the product manufacturer, if it requires that the artefact is re-worked for example, or it may be paid by the customer, if the performance of the artefact is slightly below optimum, or it may be paid by some other part of society - for example if a vehicle's engine produces slightly higher levels of noxious gases than necessary. The quality loss function gives the loss incurred by society as a whole, expressed as a monetary value, as a function of the design characteristic, and is of the form shown in Figure 3-7.

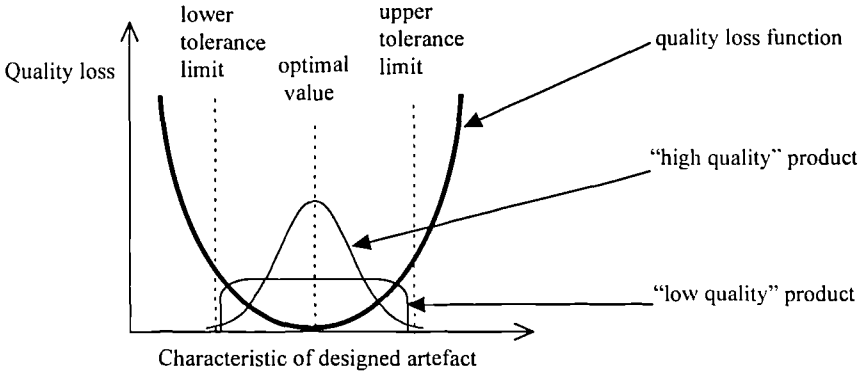


Figure 3-7: Taguchi's Quality Loss Function

Taguchi observes that, because of the non-linear form of the quality loss function (usually modelled as a quadratic function), where there is stochastic variation in the measured characteristic, the expected value of the quality loss is much smaller for "high-quality" products with a sharp peak in the distribution around the optimal value, than for "low-quality" products with a more widely-spread frequency distribution. Thus the aim of the robust design method is to minimise the stochastic variations in the quality characteristics of the designed artefact - and Taguchi argues that, although the causes of such variations (termed "noise factors") are often part of the operating environment, the variations can be considerably reduced by choosing suitable values for the design parameters (termed "control factors" - see Figure 3-8).

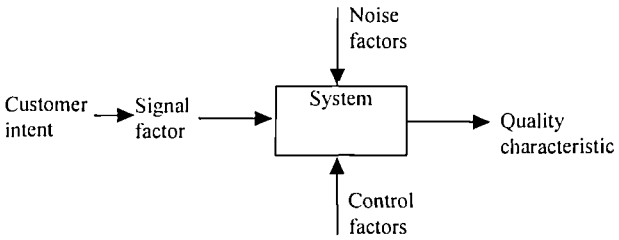


Figure 3-8: Taguchi's view of the designed artefact as a system

Taguchi divides the design process into three phases:

- *Concept design.* A system is defined which functions under an initial set of nominal conditions.
- *Parameter design.* The concept design is optimised by choosing values for the design parameters (termed "control factors") which minimise the system's sensitivity to "noise".
- *Tolerance design.* The upper and lower allowable limits on the optimised design parameter values are specified.

During the concept design phase, the superior design concept is selected from amongst the alternatives under consideration. Tools such as Pugh's method of controlled convergence (above) and Quality Function Deployment are used.

The aim of the parameter design phase is to choose nominal values for the control factors which are relatively insensitive to the causes of stochastic variations in the quality characteristic. This is achieved using a technique named Taguchi-class experimental design. Experiments are conducted where the control factors are set to different values and the resultant effect on the quality characteristic is measured - the optimal values of the control factors to minimise the variance of the quality characteristic are deduced from the results of these experiments using a statistical technique (see Appendix B for a description of Taguchi-class experimental design).

In the tolerance design phase, cost considerations are balanced against quality loss to determine the variance of the control factors from their optimal values which is allowable. Limits on noise levels may also be determined. The quality loss function is used, in conjunction with experimental design and a statistical technique called Analysis of Variance (ANOVA) is used to determine the contribution to total variation made by each noise and control factor.

3.1.7 Ullman, Dietterich and Stauffer - protocol studies

Unlike Pahl & Beitz, Hubka & Eder, Pugh and others whose main emphasis is on developing a prescriptive design method based on their experience, Ullman's earlier work with Dietterich and Stauffer concentrated more on direct, detailed observation as a means to developing a descriptive theory of the design process. He and his co-workers have documented *protocol studies* [Ullman 1987] where a group of six designers were observed during the design process, and have drawn conclusions from what they recorded. The designers were presented with open-ended mechanical design problems and were asked to verbalise their thought processes as they worked towards a solution. The designers were recorded on video tape. The studies were analysed using a state-operator-strategy paradigm, again based on a transformation model of the design process: the analysis identified changes to the state of the design, the operators the subject used to change the state and the subject's strategy for employing these changes.

Ullman and his co-workers reached several conclusions from these studies - some of which did not agree with the process models of canonical design described above. Firstly, they concluded that, rather than beginning by performing the functional design (during the conceptual design phase) and then proceeding to form design (during the embodiment design phase), as prescribed by the canonical schools, the design of function continued intermittently throughout the design process. And function is often not quantified - the analysis frequently remains qualitative throughout. Secondly, they concluded that qualitative reasoning is an essential part of the design process - in design, there is much knowledge which cannot be easily quantified. The third conclusion was that the designer's domain-specific knowledge influenced all aspects of the design - in particular that solution generation was based on adaptation of past experience, not on a systematic exploration of all scientifically possible solutions - again, in conflict with the prescribed method. Solution evaluation and choice of problem solving method were also strongly influenced by the designer's knowledge. The fourth conclusion was that designers use simulation extensively to evaluate proposed designs - whether this simulation be visual, cognitive, computational or physical. Finally, Ullman and his co-workers found that designers seek sufficient design solutions, not optimal designs (performing the process which Simon termed "satisficing",as opposed to "optimizing", and noted was common in traditional engineering design [Simon 1969]).

In [Ullman 1988], the Design Episode Accumulation Model (DEAM) of the design process is presented, which was based on an analysis of the high-level processes enacted by the subjects in the protocol studies. This model, which represents the "non-routine" design process, is based on a *design state* which is acted upon by sequences of *design operators* during *design episodes*. The design state consists of all the information about a design at a given moment in the design process and includes specifications, constraints, proposed designs under consideration, drawings, calculations etc. A design operator causes a change in state - by performing a calculation, generating a new proposed design, evaluating a proposed design or making a decision to reject or accept a proposed design.

Each episode in the design process is aimed at achieving a particular goal - and goals evolve during the design process, from being broad and abstract to being narrow and concrete. Episodes can contain sub-episodes and may be completed when:

- A goal is achieved.
- A goal is rejected and no solution found.
- A goal is suspended, and will be returned to later in the design process.

A set of goals make up a task, and a task falls into one of the three design phases identified by Pahl and Beitz - conceptual, layout or detailed.

The key points about DEAM are:

- Alternative design proposals are only considered within a particular episode - once the episode is completed, a choice is made and the alternative design proposals which arose during it are dismissed.
- The design proceeds by incremental changes to an initial conceptual design.
- There is no overall "grand plan" in the design process. A goal is pursued, and when the episode is complete, another goal is selected and pursued.

It is interesting to note that the problem-centred view of the design process implicit in the last point above is in agreement with the VDI model and in conflict with the solution-centred traditional approach.

In [Stauffer 1991] the same protocol studies are analysed for the more low-level, fundamental processes. The mechanical designer is modelled as an information processing system with short and long-term memory, a controller and operators. The short term memory forms part of the design state and the controller controls the flow of information between the short term memory and the rest of the system - see Figure 3-9.

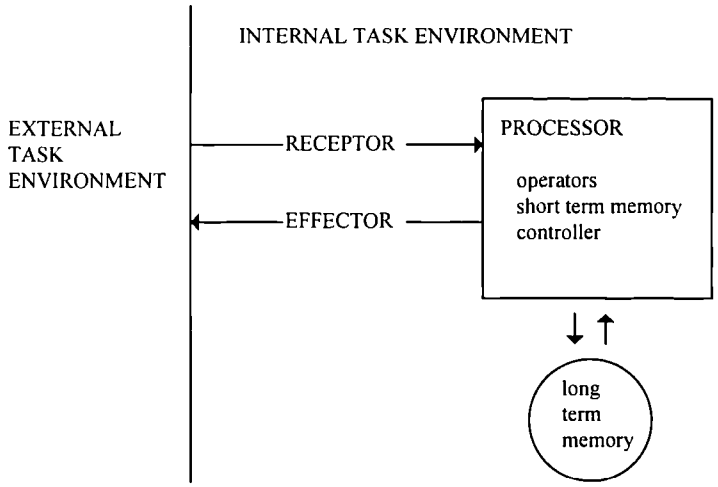


Figure 3-9: Model of the mechanical designer as an information processing system, from [Stauffer 1991]

The types of information present in the design environment are categorised as:

- **Strategies** - procedures for achieving a goal.
- **Proposals** - possible solutions.
- **Constraints** - which include the initial problem specification, limitations imposed by the designer's experience and limitations imposed by design decisions.

The operators can change the design state. Three types of operator are identified:

- **Generate** operators, which introduce new information into the short-term memory. These are divided into the *select* operator and the *create* operator. The select operator selects information from the problem statement, from the designer's specific previous experience or from the designer's general domain knowledge. The create operator seemingly "spontaneously" creates new information.
- **Evaluate** operators, which relate generated information in order to make a decision. These are divided into the *simulate*, *compare* and *calculate* operators. The simulate operator converts information to a different level of abstraction. The compare operator determines compatibility between pieces of information (e.g. a proposal and a constraint). The calculate operator infers new information by combining existing information - where the information combined may be quantitative or qualitative.
- **Decision** operators, which are divided into *accept*, *reject*, *suspend*, *refine* and *patch*. If the evaluation result is acceptable, the accept operator adds information to the design state and terminates the decision. If unacceptable, the reject operator terminates the decision. The suspend operator terminates a decision without coming to a definite conclusion. The refine operator modifies the information to make it more specific and less abstract. The patch operator adds to or combines information without altering its level of abstraction.

A sequence of operators, applied to solve a particular problem, is termed a *local method*. Stauffer and co-workers found that four local methods in particular were used repeatedly during the design process:

1. Generate and test - a proposal is repeatedly generated and tested against a constraint until a proposal is found which satisfies the constraint.
2. Generate and improve - a proposal is generated, then it is repeatedly patched or refined and then matched against a constraint until a better match is obtained.
3. Means-end-analysis - a proposal is generated, evaluated against a constraint and then the proposal is patched or refined so as to remove the difference between the present state and the desired state (constraint satisfaction is the desired state).
4. Deductive thinking - a proposal is generated and evaluated against a constraint, and then a new proposal and constraint are calculated (i.e. deduced or inferred) such that the proposal satisfies the constraint.

In all of the above local methods, proposals were generated by *selection*, as opposed to spontaneous generation by the *create* operator. It was found that 95% of local methods used in the protocol study fell into these four categories, and that this was made up as follows:

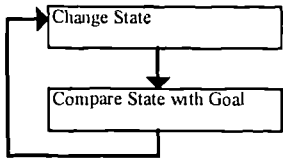
generate and test	23%
generate and improve	8%
means-end-analysis	36%
deductive thinking	33%

3.1.8 Ullman

In his textbook "The Mechanical Design Process" [Ullman 1992], Ullman divides mechanical design problems into the following categories:

1. Selection design - for example choosing a bearing from a catalogue.
2. Configuration design - determining how to assemble existing components into a product.
3. Parametric design - finding values of attributes which characterise the artefact.
4. Original design - developing a new process or artefact.
5. Redesign (which may be routine or may not) - modifying an existing product to meet new requirements.

Categories 4. and 5. are similar to Pahl & Beitz's "original design" and "adaptive design", although both categories will contain some elements which correspond to 1. - 3. Pahl & Beitz's "variant design", where the solution principle and task remain the same but some design parameters are varied, will contain elements which correspond to categories 1. and 3. Ullman's categories 1. - 3. characterise design problems at a lower level of granularity than Pahl & Beitz and his own categories 4. and 5. As in Ullman's earlier work on the DEAM model, the design process is viewed as an incremental sequence of changes to the *design state*, punctuated by *design decisions*. Ullman considers that there are two possible views of the process driving these changes (as shown in Figure 3-10):



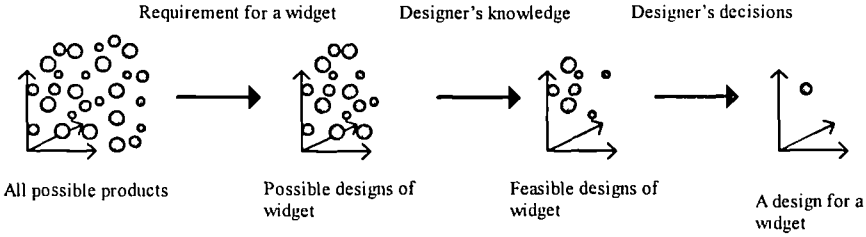


Figure 3-10: Design as a goal-driven process or as addition of constraints to search space

1. The design process may be driven by a continual comparison of the design state with a well-defined goal, with the state being continually changed to bring it closer to the goal. This is only possible where the goal is initially known and clearly defined and the relationship between the design state and the goal is clearly understood. Ullman suggests that some simple selection and configuration problems can be solved in this way; clearly, any parametric problem which can be solved by optimisation also fits this process model.

2. The design process may alternatively be viewed as the continual addition of constraints, gradually narrowing a search space of possible product designs. Conceptually, one begins with the space of all possible product designs. This search space is then constrained; initially by the problem requirements, then by the designer's knowledge about previous designs and about the problem domain. The designer's decisions (design choices) then further reduce the search space. This process continues until only one solution is left.

This latter view of design as a continual reduction of the search space clearly does not model the process as enacted by a human designer. Unless one is simply considering parametric design, the design state is unlikely to ever include all possible designs of the artefact; rather than constraining a large search space, the designer's knowledge could alternatively be viewed as largely defining the search space.

3.1.9 Esterline et al.

Esterline and co-workers tried to study problem formulation in mechanical design but concluded that it could not be separated from the design process itself. Expert, sophisticated designers were asked to formulate a problem in the kinematic design of mechanisms. They were unable to formulate the problem without performing some design: illustrating Cross's point, quoted above, that formulations of design problems are often solution dependent. In [Esterline 1988] the authors present a general formulation of the design process for mechanical design, based on their protocol study.

3.1.10 Cross

In [Cross 1994] the author proposes a variation on the canonical models of the design process which partially acknowledges the shortcomings of the Pahl & Beitz serial design-phase model and of the VDI problem-centred decomposition. The central driving force behind the process, shown in Figure 3-11, is an anti-clockwise spiral, where the problem is decomposed into sub-problems which are solved and then combined, as in the VDI model. Unlike the VDI model however, in Cross's model there are symmetrical, two-way relationships between problem and solution and between sub-problem and sub-solution. However, Cross's prescriptive model is still essentially serial with iterations, rather than concurrent.

In [Acar 1996], the author points out that design "iteration" is not simply repetition since, when a design stage is re-visited, progress has been made by the acquisition of experience. Perhaps "design recursion" would be a better term? Acar suggests a "triple-helix" as a visualisation model of a prescribed design process where the three strands are specification, conceptual design and embodiment design and the major axis is time. Thus specification and conceptual design should be revisited with increasing levels of experience. Each triple helix represents a system or sub-system and is cocooned in its "environment", which may include other

systems/sub-systems. Acar proposes that all systems and sub-systems should be developed concurrently but doesn't discuss how the necessary synchronisation between design phases of dependent systems can be achieved.

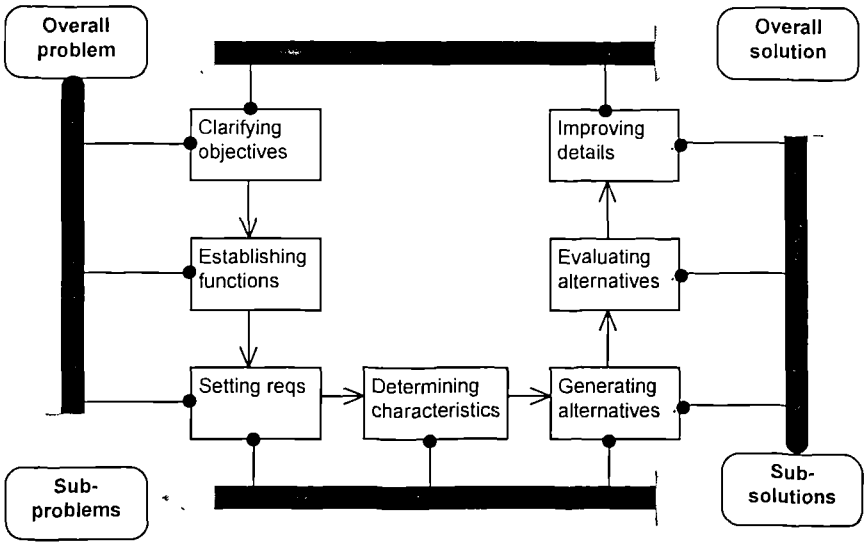


Figure 3-11: Cross's spiral model of the design process, from [Cross 1994]

3.1.11 Ohsuga

In his paper describing the requirements for an "intelligent" CAD system [Ohsuga 1989] (one which incorporates knowledge-based capabilities), Ohsuga considers the design process which such a system must be able to represent. He argues that there is no standard design process; it depends on the individual designer. Ohsuga however describes a typical design process which is "widely used in many engineering domains", shown in Figure 3-12.

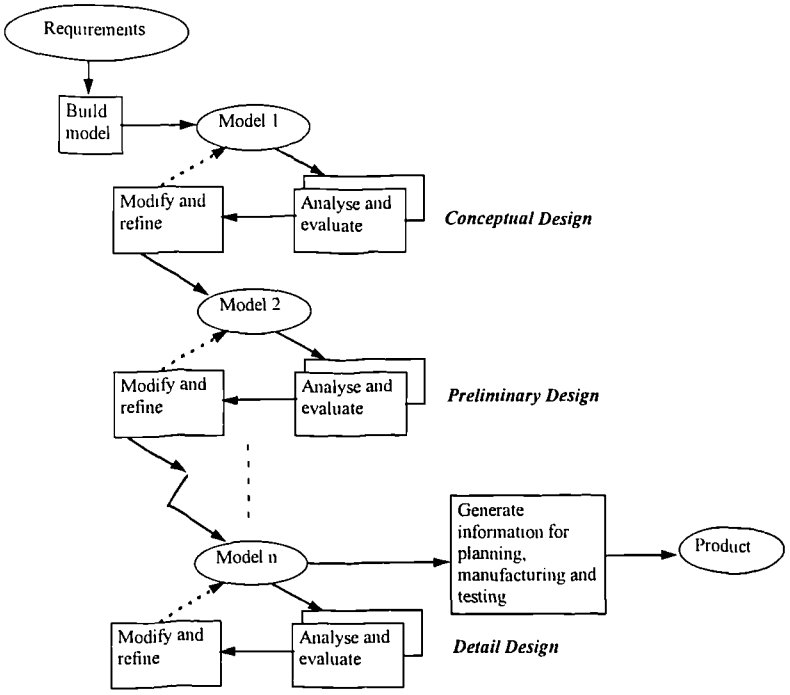


Figure 3-12: Ohsuga's model of a typical design process

In Ohsuga's view of the design process, an *object model* is subjected to a sequence of *transformations*. A distinction is made between *equivalent* and *non-equivalent* transformations. An equivalent transformation is one which does not modify the artefact model - the result of analysis, it is deterministic in that the transformed model can be entirely determined from the original one. A non-equivalent transformation is one where information is added to the artefact model - the result of synthesis, it is non-deterministic in that it relies on search to provide new information.

An initial object model is tentatively built; this will be an imprecise model, at a low level of refinement. The model is analysed and its functionality is evaluated; initially using rough or qualitative estimation methods. The model is analysed and evaluated from several different viewpoints - for example, structural strength, aerodynamics, etc. If the model doesn't meet the requirements, it is modified and the analysis and evaluation are repeated. When the model is deemed to meet requirements at its current level of refinement, the designer takes it to the next level by the addition of information. Then the process of repeated analysis and evaluation followed by modification is repeated but at ever higher levels of refinement.

3.1.12 Bond & Ricci

Bond & Ricci's work on the collaborative design process [Bond 1992] is based on observations from the aircraft industry, although less formally observed than the protocol studies. Their view of a collaborating design team is of many specialists, each with their own model/s, communicating via a shared vocabulary (Figure 3-13).

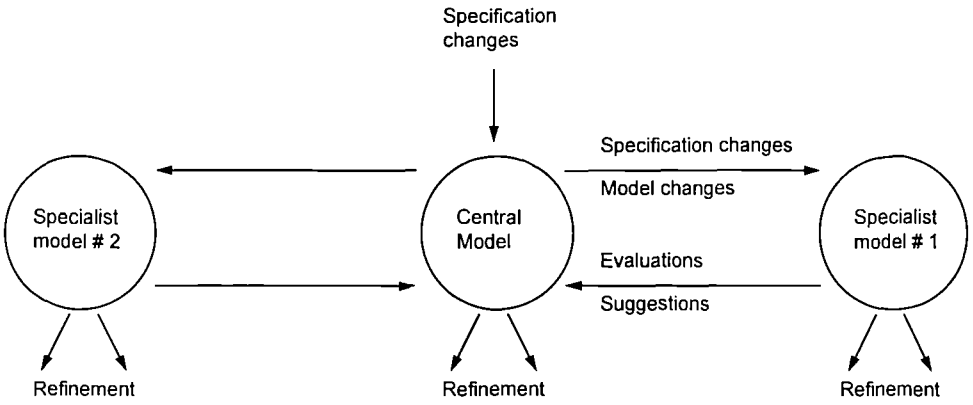


Figure 3-13: The design process as co-ordinated refinement of models, from [Bond 1992]

As the design proceeds, the models are gradually (and concurrently) *refined* - this process is co-ordinated by the use of a central model (e.g. a CAD model or drawings), with which each specialist must keep his own model/s in step and by *negotiation* between the experts at each *commitment step*. The idea of refinement is best explained by some examples:

- Moving from an approximate volume (e.g. cuboid) to more detailed geometry.
- Moving from an approximate numerical estimate (e.g. interval) to a more exact one.
- Replacement of one (or possibly more) abstract element/s with multiple more detailed elements.
- Addition of elements.

At the end of each commitment step, a set of joint commitments is made by all the specialists. Each specialist (or "agent"), generates (subject to negotiation) *public* results which contribute to the central model and are thus used by all the agents, and *private* results which are only used within their own specialised model. Thus, at the end of a step, each agent must provide their public results to a sufficient level of accuracy (or degree of certainty) such that all of the agents can complete the next step. The results might, for example, be represented as intervals.

3.1.13 McMahon et al.

The design process model proposed by McMahon *et al.* [McMahon 1995] is neither prescriptive, nor the direct result of protocol studies, although it utilises results from both types of model. It is suggested as a possible basis for the "shared memory" proposed by Konda *et al.* [Konda 1992] and at a lower level, as a basis for design automation. The theory behind Konda's concept of "shared memory" is that, although design includes many disciplines, it is not in itself a scientific discipline; that the aim of devising prescriptive context-free design methods (such as those described above) which will improve design practice can never be achieved; and what is required instead is records of previous design cases, which have a shared meaning across disciplines (and also between departments and individuals).

In common with Hubka & Eder, Ullman and others and in particular Bond & Ricci and Ohsuga, McMahon and co-workers regard the design process as a sequence of transformations between states of information models describing the design. Three types of information model for the design itself are considered: Firstly, models representing *explicit attributes*; these are the objective, measurable, physical properties of the artefact taken in isolation, such as dimensions. Secondly, models representing *implicit attributes*; these represent the evaluation of the design according to economic, technical or other criteria and derive from the interaction of the artefact with its environment. Thirdly, *auxiliary models* contain the information required to deduce the implicit from the explicit attributes. In addition to these three types of design model, there are also information models of the functional requirements, the constraints on the designer and the environmental influences on the design (typically the loading).

Design is viewed as several interdependent, concurrent processes (see Figure 3-14).

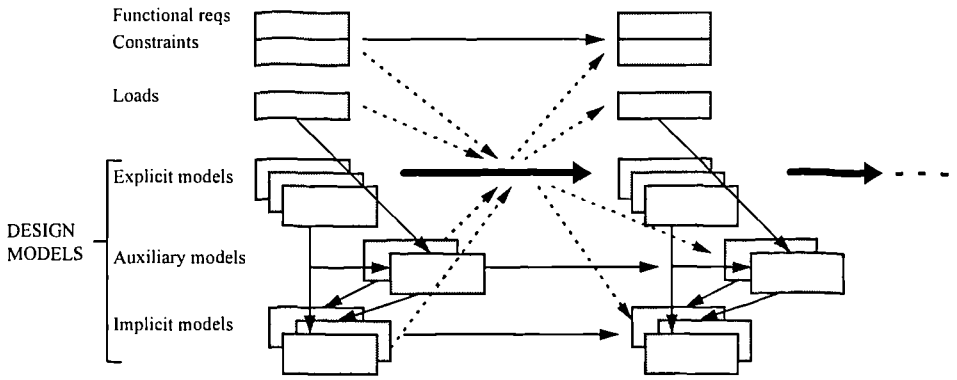


Figure 3-14: Interactions Between Design Representations, from [McMahon 1995]

In the central thread, the explicit models are developed and refined as the design emerges; functional requirements and constraints may also be refined as the design proceeds (as noted by Cross, Pahl & Beitz

etc.). Concurrently, quantitative *evaluations* of the emerging design augment and develop the *implicit* models, often requiring the development of *auxiliary* models - subjective judgements may also augment the implicit models. Examples of auxiliary models, used to deduce implicit attributes, might include a Finite Element Analysis model for stress analysis, or a parametric cost estimation model. The notation of Petri nets [Petersen 1981] is used to model this set of parallel processes.

A property of the design process, noted by McMahon *et al.*, Bond & Ricci and others, and of particular interest from the perspective of uncertainty in design, is that there may be several alternative paths through the design process, each leading to the same implicit attribute but each with a different cost (in terms of time or input information required) and each yielding a different precision (or degree of uncertainty) in the result. Each of these alternative paths will use a different auxiliary model.

> *Throughout [the design negotiation] process, time and cost of analysis play an important role as to the depth of analysis actually undertaken and the number of iterations allowed. In the end, these two factors are what closes off further development and the development community settles down into a 'make this work' situation.*
> [Bond 1992]

In [McMahon 1994], the process of *historical* development of established designs is also considered in terms of implicit and explicit design attributes. It is proposed that modes of change in such designs fall into one of the following five categories - where the first four are incremental and evolutionary:

- Design parameter space exploration.
- Improvements in understanding of design attribute relationships.
- Changes in the product design specification.
- Modification of the feasible design space.
- Adoption of a new design principle.

3.1.14 Ward and Seering: Set-Based Concurrent Engineering

In [Sohlenius 1992], one of the major challenges of CE is defined as "how to avoid changes at later stages where incurred costs are high". The traditional CE approach is to try to improve the quality of early decisions; but an alternative approach, advocated by Ward *et al.*, is to attempt to avoid making early decisions. Rather than iteratively developing a single design solution (what Ward refers to as point-based design - moving a single design point through design space), many alternative design solutions are pursued simultaneously, with the choice between them being made as late as possible.

> *Designers explicitly communicate and think about sets of design alternatives at both conceptual and parametric levels.*
> [Ward 1995]

Thus the aim is to delay decisions, encourage "ambiguous" communication between team-members (for example using ranges in specifications to suppliers) and to create many prototypes. The design process is seen as a process of parallel set narrowing.

The concepts of set-based CE grew from Ward and Seering's earlier work (described in Appendix B) where they developed a calculus of labelled intervals to support an automated design tool (their mechanical design "compiler") which made inferences concerning sets of components or attribute values, rather than enumerating all possible combinations of components in a conventional optimisation. According to Ward and Seering [Ward 1995] and Sobek [Sobek 1996], Toyota (who were also one of the pioneers of QFD in the 1970s) currently use the set-based CE paradigm to great effect in their design process.

Toyota explore more concept variants than their competition, creating more small-scale clay models for example. They delay fixing dimensions as long as possible and also delay releasing specifications to their suppliers - although approximate (or target) specifications may be released. Many more sub-system prototypes, of different designs, are built at Toyota than at competing automotive design companies. The importance of rapid prototyping to support this approach is clear. It is interesting to note that there are aspects of Toyota's corporate culture which make this approach more likely to succeed; particularly, their close and long-term relationship with their suppliers (typical of a Japanese company) and the very high level of engineering skills within the company, engendered by lengthy on-the-job education and high expenditure on training.

Many design authors recommend exploration of as many design concepts as possible - "you need as many ideas as you can possibly generate - single solutions are usually a disaster" [Pugh 1991, p. 69]. However, this is not the same as the idea of a shared and ambiguous (i.e. uncertain) design model. Although seemingly wasteful, in that many designs which are never actually manufactured are taken to the prototype stage, the set-based approach has distinct advantages over the point-based one. Given the ill-defined nature of most design problems, and the cyclic dependencies which often exist between design decisions, it should generate "better" design solutions than point-based design. By encouraging ambiguous communication, it provides a means of implementing CE but without requiring such intense communication between team members - avoiding the problem of endless meetings leaving designers with no time to design, for example.

3.1.15 Conclusions

Taken to an extreme, the design process prescribed by Pahl & Beitz and others suggests that, ideally, all scientifically possible potential design solutions *should* be generated and quantitatively evaluated - solving a design problem as though it were a natural science problem, by generating conjectures which are then evaluated and, if found to be incorrect, refuted. Implicit in the idea that design practice *can* be modelled in this way (though possibly at a very low-level) are the principles of Artificial Intelligence.

However, the work of Ullman *et al.* and others shows that designers do not work in this way - rather, they prefer to pursue a single design concept and will patch and repair the original idea rather than generate new alternatives. Also, Ullman *et al.* found, in their work on fundamental, low-level processes, that design proposals are usually *selected* from the problem statement, from the designer's specific previous experience or from the designer's general domain knowledge rather than being *created* in a more spontaneous sense.

Finger and Dixon [Finger 1989] quote references from as long ago as 1961 indicating that:

> *Designers re-use familiar solutions and will not explore alternatives or innovative ideas unless their new design fails badly and cannot be salvaged.*
> [Marples 1961]

Designers' knowledge and historical experience is used to reduce the search space so that a satisfactory, but not necessarily optimal, solution can be found within the limited time and budget available. Designers re-use familiar solutions in order to reduce uncertainty and risk, and this re-use of familiar solutions can be regarded as a strength, not a failing. A designer who seeks a "better" solution so assiduously that the project deadline comes and goes will not be regarded as successful. In particular, it could be argued that even conceptually radical designs are based on variations of historical blue-prints, not on an exhaustive search of the solution space.

As a prescriptive model, set-based concurrent engineering is an attempt to avoid the highly sub-optimum designs which could result from this observed tendency to patch and repair an initial solution, by increasing the level of uncertainty in the early stages of the design process.

Considering how the "degree of uncertainty" in a design model varies against time during the design process, the views expressed in the preceding paragraph indicates that a "sawtooth" curve in Figure 3-15 may in fact provide a more accurate model of current practice than the monotonic curve shown in Figure 3-1.

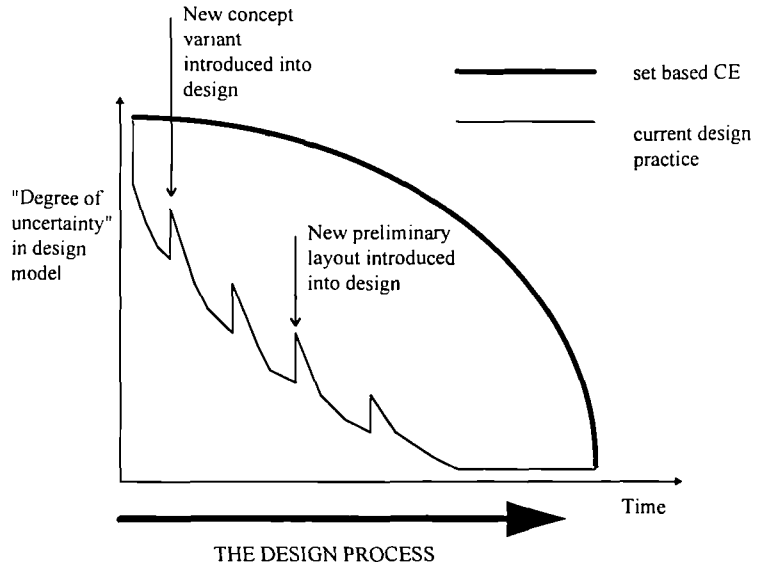


Figure 3-15: Evolution of uncertainty during the design process

For example a new (and thus highly uncertain) concept variant may be introduced only after the concept/s initially selected by the designer have been sufficiently developed so as to prove their inadequacy. Similarly, new form/layout solutions may be introduced after quite detailed exploration of (and corresponding reduction of uncertainty in) familiar solutions proves them inadequate. The smooth second curve shown compares the evolution of uncertainty in a set-based CE environment.

Having reviewed a variety of design process models, we are now in a position to state more clearly what is meant in this thesis by *early design*. Using the terminology of Pahl & Beitz for the design phases, we would include clarification, conceptual design and the early stages of embodiment design but would exclude the

detail design phase. The type of design problems faced during early design will include Ullman's selection design (for example, selection of a component from a catalogue), configuration design (assembling existing components into a product) and also early parametric design.

3.2 Types of Uncertainty Present in Early Design

The views of the design process presented in the preceding section suggest a variety of types of uncertain information which are present during the design process. In this section, the following question is addressed:

"How can we classify the types of uncertain information which are important during the early phases of the design process?"

Before proposing an answer, we briefly review current research relating to modelling uncertainty in early design in order to support design decision-making. Butler *et al.* [Butler 1995] have recently produced a short review in this area and Antonsson and Otto [Antonsson 1995] a slightly longer one. The aim is to identify some of the types of uncertain information which are currently considered important in early design, and also to present the taxonomies or classifications which have been proposed by other researchers.

3.2.1 Ullman et al.

In [Ullman 1995] Ullman and D'Ambrosio consider how best to help support designers in evaluating design alternatives, and propose a taxonomy for engineering design decisions and decision support systems. This is of particular interest, as they identify several types of uncertain information about a design decision. The authors observe that many design decisions (especially early ones) are based on information which is sparse, qualitative or abstract and conflicting. As noted earlier, it is simply not feasible to collect the information required to make all decisions on a quantitative basis. The taxonomy suggested is shown in Figure 3-16.

The intention is that all design decision problems and also all decision support systems can be classified by assigning a value to each of the 11 fields identified. A decision problem is characterised by the decision space (described by the alternatives and the criteria), a preference model (usually described by an objective function or a utility function) and a belief model (representing the designers' knowledge about the alternatives and confidence in them satisfying the criteria). Ullman and D'Ambrosio observe that a typical engineering design decision problem may involve incompleteness and abstraction in the decision-space, an inconsistent preference model and an incomplete belief model. They seem to consider that both the decision space and also the preference and belief models in Figure 3-16 should be considered as parts of the design state. Ullman and D'Ambrosio regard a variety of models for uncertain design information as decision support systems and categorise them using their taxonomy; including probabilistic design, optimisation, CPM/PERT, decision trees, Pugh's method and utility theory. They conclude that existing tools do not address many of the core requirements for engineering decision support.

Structure of the decision problem	Decision space set of all possible outcomes - defined by the alternatives (variables) and the criteria for their evaluation	1. Problem completeness Are all the alternatives known? Are all the criteria known?
		2. Quality Are alternatives and criteria abstract (qualitative) or refined (quantitative?)
		3. Determinism Are variables represented as point values or as distributions?
	Preference model represents the goal of the problem or the value of each potential outcome.	4. Objective function determines how well each alternative meets the criteria. Can it be quantified (using parameters of alternatives) or does it rely on judgement?
		5. Consistency Are there conflicting views amongst the team over the relative importance of each criterion or the relative goodness of each alternative?
		6. Comparison basis Are alternatives evaluated absolutely against criteria, or relative to each other (for each criterion)?
	Belief model represents the designers knowledge about an alternative or criterion, and confidence that an alternative will meet a criterion	7. Dimension 0 = None, 1 = Knowledge OR Confidence, 2 = Knowledge AND Confidence
		8. Belief completeness Have all the designers expressed their confidence level in all <alternative, criterion> pairs?
Focus	9. Problem focus Is the focus of this decision problem the product itself (the artefact under design) or is the focus one of the processes supporting the design, manufacture or distribution of the product?	
Range	10. Range of issue independence Can this decision be made independently of other issues? Or does it depend upon decisions made elsewhere? Or is it part of a complex interdepedency network?	
Support	11. Level of support offered to (or required by) the decision-makers may be simply representation of the decision space, or outcome determination or decision analysis	

Figure 3-16: A taxonomy for engineering decision support, from [Ullman 1995]

In [Herling 1995] a decision support system (a methodology and software tool) is presented based on this formulation, which models, for a team of designers:

- The design alternatives.
- The criteria against which the alternatives are to be evaluated.
- A designer's level of knowledge about an <alternative, criterion> pair.
- A designer's confidence that an alternative will satisfy a criterion.

The information provided to the tool by the designers may be sparse - i.e. it is not necessary for all designers to evaluate all alternatives against all criteria. The tool uses a probabilistic model to provide a point-valued "satisfaction indicator" for each of the alternatives.

In Ullman, D'Ambrosio and Herling's work, the types of uncertainty which are considered important include inconsistency, abstraction and incompleteness as well as that which may be represented by non-deterministic variables. Whilst the preference and belief models are clearly a part of the description of a decision problem, it could be suggested that they are not necessarily a part of the design model.

3.2.2 McMahon

The definitions of implicit and explicit design attributes proposed by McMahon *et al.* were given earlier, in Section 3.1.13. McMahon points out in [McMahon 1994] that the presence of uncertainty reduces the range

of explicit design attribute values from which the designer may choose. The following ways in which uncertainty may thus reduce the design space are identified:

1. Stochastic variations in the explicit attributes. The values of dimensions, for example, are subject to manufacturing variations.

2. Uncertainty in the limiting relationships which help to define the feasible values for explicit attributes. For example, the surface finish that a manufacturing process could achieve.

3. Uncertainty in the relationships between implicit and explicit attributes and loads. The first order reliability method may be used for example to relate probability of failure to dimensional/material values and loads, but the assumption of linearity introduces uncertainty into the relationship.

The important new type of uncertainty which is introduced here is uncertainty in relationships as opposed to attribute values.

3.2.3 Thurston et al.

Like Ullman *et al.* above, the work of Thurston and co-authors has addressed the problem of design evaluation (to support a decision between design alternatives) under circumstances where there are multiple incommensurate and related implicit design attributes and where the values of these attributes are subject to uncertainty. In [Thurston 1991a], a probabilistic representation is used for the implicit design attributes and utility theory is applied to provide support for a decision between them (see the section entitled Decision Support Techniques in Appendix B). Uncertain values for multiple implicit attributes - for example cost - are represented using Beta probability density functions and the attitude of the design engineer to risk and to trade-offs between attribute values is incorporated into a utility function.

In [Thurston 1991b] the utility function approach is used to automate the choice of optimal values for explicit design attributes. A structural engineering application is presented for example in [Locasio 1993] where a building is subject to vibration. Two implicit attributes (cost and occupant-discomfort level) are expressed as a function of two explicit design attributes (the stiffness of each of two columns). A multi-attribute utility function is then constructed which represents the design engineer's preferences regarding cost and occupant discomfort, and his willingness to make trade-offs between the two. A non-linear optimisation method is then applied to arrive at the optimal values for the stiffness of the columns.

In [Thurston 1994] and [Tian 1994], these type of utility functions have been incorporated into an expert system which reflects the subjective attitude of a particular expert towards risk. [Tian 1994] describes an expert system which addresses the issue of risk arising from uncertainty in manufacturing cost estimation and other design attributes including weight. A heuristic rule base containing objective technical information is used to determine a set of technically feasible alternatives to a given design problem (a material selection problem for an automotive bumper in this case). The uncertain values for cost, weight etc. are represented by Beta distributions, elicited as the 5th, 50th and 95th percentiles plus absolute bounds (not as max, min and

mode as in PERT) for each feasible alternative. The subjectively determined utility function, obtained using certainty-equivalence questions as usual, is then used to rank the alternatives according to expected utility.

In [Thurston 1992] a fuzzy weighted average approach to this problem is taken; the implicit attribute values are represented by fuzzy numbers "low", "medium", "high" etc. and the weightings are also assigned fuzzy values (see Chapter 5, Section 5.2 "The Choice of Representations"). The types of uncertainty considered to be most important in this work are thus numerical uncertainty in explicit and implicit attribute values, linguistic vagueness in implicit attribute values and in Ullman's preference model, and the existence of design alternatives.

3.2.4 Allen, Mistree et al.

For more than a decade design research has been conducted at the Georgia Institute of Technology in which design problems have been formulated as Decision Support Problems (DSPs) of various categories and solved using methods which include linear and non-linear programming techniques; in a *selection DSP* an alternative is chosen from several by rating multiple attributes, in a *compromise DSP* an alternative is improved by optimising design variables, a *hierarchical DSP* involves combined simultaneous solution of compromise and selection DSPs, and a *conditional DSP* involves explicitly taking uncertainty into account in the decision [Shupe 1987]. In [Allen 1992] , uncertainty was incorporated into the compromise DSP problem using fuzzy set theory. The constraints of the optimisation and its goals were represented using fuzzy numbers (see Appendix B), but the solution obtained remained crisp. The important types of uncertainty introduced here are therefore vagueness in the design constraints (similar to McMahon's uncertainty in limiting relationships) and vagueness in the design goals (the functional requirements).

In [Peplinski 1996] a software tool called FLAME (Function Level of Abstraction Manufacturability Evaluator) is presented which embodies a different (simpler) approach to evaluation of multiple attributes under uncertainty from that of Thurston *et al.* The problem of choosing suitable materials and manufacturing processes to solve a design problem from a database of available solutions is formulated as a "heuristic selection DSP". A product model is incrementally developed at three levels of abstraction - initially at a functional level, then at a conceptual level when the solution concept has been identified and finally at a component level. At each level, pieces of geometric information (specifications) are attached to entities within the product model as text strings, numerical values, etc. At the functional and conceptual level, the software tool simply identifies whether or not the database contains any combinations of material and manufacturing process which are compatible with the design specifications - at the functional level abstract descriptions of materials and manufacturing processes are used, becoming more specific at the concept and then component levels. Thus a set of technically feasible alternatives are identified and gradually narrowed as the product model is refined. Then, at the component level, each of the set of alternatives is evaluated according to their economic efficiency where closed interval representations may be used to represent the economic specifications. Not only manufacturing cost, but other implicit attributes such as product quality and environmental impact are modelled, but the values all are normalised to be measured in currency units. The multiple criteria are combined in a linear merit function (a weighted average is taken) and each alternative is thus assigned a range (max and min value) of merit function values. These are converted to a range of rankings - the maximum ranking for alternative A is achieved when the merit function for A takes

its maximum value and the merit functions for the other alternatives take their minimum values, and vice versa. A decision can then be made which explicitly takes the degree of uncertainty into account.

FLAME combines several features which are of particular interest here: support for a design process based on refinement as reduction in abstraction as well as addition of information, re-use of a database of existing resources and the modelling of uncertainty in engineering design. The important types of uncertainty introduced are abstraction and numerical uncertainty in an implicit attribute value (e.g. cost).

3.2.5 Antonsson, Otto, Wood et al.

Wood and Antonsson's "method of imprecision" based on using fuzzy numbers to express designer preferences is described in the section concerning the Extension Principle in Appendix B. In [Antonsson 1995], a review of available methods for incorporating imprecision in engineering design decision-making is presented; the methods discussed are combination functions, utility theory, fuzzy sets, optimisation, matrix methods, probability methods, necessity methods and fuzzy design methods.

In [Wood 1990] [Antonsson 1995] etc., the authors distinguish between *stochastic uncertainty* and *design imprecision*. *Stochastic uncertainty* arises from uncontrolled stochastic variations which are represented using probability theory (e.g. engineering manufacturing variations due to tooling non-repeatability, material property variations etc.). Techniques for determining the risk of a design failing in the face of such uncertainty include Taguchi's method, utility theory etc. This is distinguished from *imprecision* which, it is suggested, is usually taken to mean uncertainty in choosing among alternatives. An imprecise variable may be discrete (e.g. representing the choice between two totally different types of suspension system for a vehicle) or may be continuous (e.g. representing the choice of a dimension of a component), but it will always be fully determined when the design is complete.

In [Otto 1993], Otto and Antonsson consider *design imprecision* and *stochastic uncertainty* as above in addition to *possibilistic* variables and *necessity* requirements. Here, a possibilistic variable is one which is free to take any of several (or a range) of values. An example would be a parameter whose value is determined by the user such as the seat position in a car, or a parameter whose value is adjusted after the manufacturing process is complete to "tune" or improve the product performance. A necessity requirement means that a parameter must satisfy a range of values. An example would be the range of speeds at which a motor design must operate (this corresponds to Ward and Seering's interval label "Operating region", see Section B.3.1.1 "Interval Analysis" in Appendix B). These concepts of possibility and necessity do not strictly represent uncertainty in the design model, simply ranges of values. A perfect model which predicted exactly and correctly all aspects of the behaviour of an artefact would still include possibilistic variables. A necessity requirement represents a constraint, and is only uncertain if the constraint is uncertain. Therefore possibilistic variables and necessity requirements are excluded from the proposed classification which follows.

3.2.6 A Proposed Classification

Let us return to the question posed at the start of this section:

"How can we classify the types of uncertain information which are important during the early phases of the design process?"

It is helpful to separate this question into the dual considerations:

1. What is the subject of the information?
2. What is the nature of the uncertainty?

The design process models described in the preceding section are helpful in identifying the subjects, but we now need to define some terms in order to discuss the nature of the uncertainty. So far, in this thesis, the term "uncertainty" has been used in its general, non-technical sense:

> *The quality of being uncertain in respect of duration, continuance, occurrence, etc.*
> [Shorter Oxford English Dictionary]

As mentioned above, Antonsson *et al.* consider *uncertainty* to refer to uncontrolled stochastic variations and distinguish it from *imprecision* which is under the control of the designer. In the classification proposed below, this distinction is not made between lack of certainty which is under the control of the designer and that which is not, because it depends upon the role of the participant in the design process. How should one consider lack of certainty which is under the control of the customer? or the marketing department? We concentrate on the nature and subject of the information because this gives us a classification which has the same meaning to all participants in the product design process.

In the proposed classification, information may lack certainty because it is:

- Linguistically vague (for example, the product specification for a new motorised bicycle may include a "small" power pack with "attractive styling").
- Numerically imprecise or uncertain (for example, "weighing between 10 and 20 pounds").
- Categorically imprecise or uncertain (for example, "the pack will either be placed on the back or under the saddle").
- Incomplete (for example, information concerning the power pack fixings may be missing altogether).
- Abstract (for example, "power pack" is more abstract than "nickel-cadmium battery" or "fly-wheel battery").
- Inconsistent (for example, the target weight is inconsistent with the required power output).

The distinction which is made here between numerical uncertainty and categorical uncertainty is similar to Rowe's dichotomy of measurement and descriptive (or taxonomic) uncertainty [Rowe 1977]. The terms are not identical because abstraction and linguistic vagueness, as well as categorical uncertainty, correspond to

an absence of information relating to the *identity* of design attributes, though numerical uncertainty does correspond to lack of information concerning the *values* of the attributes. The representations for these types of uncertainty are discussed later, in Chapter 5, and described in more detail in Appendix B.

The matrix shown in Table 1 proposes a two-dimensional classification for the types of uncertain information which are important during early design, in terms of the *nature* of the uncertainty and its *subject.* The *nature* of the uncertainty is given above. The *subjects* used in the classification are based on the information models identified by McMahon *et al.* (see Section 3.1.13 and Figure 3-14). The white boxes contain types of uncertain information which the risk model should be able to represent, the grey boxes represent types of information which are important in early design but are not a part of the requirements for a risk model, and the black boxes represent types of information which are not considered to be important in early design:

Nature⇒ Subject ⇓	Vague	Numerically Imprecise/ Uncertain	Categorically Imprecise/ Uncertain	Incomplete	Abstract	Inconsistent
Design goals	1.	2.		3.	4.	5.
Constraints		6.		7.		8.
Environment of the design.	9. linguistically vague specifications	10. imprecisely known loads, sales volumes,...	11. alternative future scenarios within which the design must function	12. the description of the environment is always incomplete	13.	14.
Explicit models	15. linguistically vague design choices	16. continuous design choices 17. stochastic variations due to manufacturing tolerances,...	18. design choices between alternatives	19. parts of the design not yet completed	20. designer has specified general category of artefact process/..	21. particularly in a team based environment
Auxiliary models (which generate implicit attribute values)		22. imprecision introduced by the use of heuristics and approximate methods	23. alternative models available			

Table 1: Subject and nature of types of uncertainty which are important in early design

There is a general agreement (Cross, Pahl&Beitz, & others) that the product specification will be developed during the design process. Information in the product specification concerning **design goals** (i.e. functional requirements) may be linguistically vague and numerically imprecise (cells [1 - 2] in Table 1); it is unlikely that there will be uncertainty between different categories of design goal, but incomplete, abstract and inconsistent information is certainly likely to be present [3 - 5]. Information in the product specification concerning the **constraints** within which the goals are to be achieved may in principle be linguistically vague, but in practice it is more helpful to consider such information as a modification to the design goals. A textual description of design goals, subject to numerically specified constraints is conventional. The constraints may well be numerically imprecise, incomplete or inconsistent [6 - 8], but uncertainty between different categories of constraint is unlikely as is uncertainty due to abstract constraint definition. There may also be information in the product specification which is only meaningful in the context of a particular design

solution, but this is not necessarily uncertain. It may be certain if the envisaged solution is adopted, and will be irrelevant otherwise.

The exact conditions of use are not known and nor, precisely, are the externally determined costs, or volume of sales for example. Such information, concerning the **environment of the design** (i.e. within which the design must function), is subject to all types of uncertainty [9 - 14]. The same is true of the explicit attributes (such as dimensions) which comprise the **explicit models**. The designer may well initially specify explicit attribute values in a linguistically vague fashion [15], or a numerically imprecise one [16]; the source of both types of uncertainty being design choices which have not yet been made. It is also often not known whether the manufacturing process can produce the desired explicit attribute values - for example the stochastic variations caused by manufacturing tolerances give rise to explicit attribute uncertainty [17]. This corresponds to Wood and Antonsson's stochastic uncertainty. There may also be a choice between alternative designs, each of which has been included in the model, which must be evaluated [18]. The designer may not yet have chosen the exact type for a manufacturing process, a component, etc. but the general category may be known [20]. In a team-based design environment there is certainly room for inconsistency in the explicit design model/s [21].

There are two important types of uncertainty which are manifest in **auxiliary models** (e.g. FEA models or algorithmic cost models). The relationship between the explicit and implicit attributes can never be known precisely and thus the model itself introduces some uncertainty [22]. There may also be several alternative relationships, each giving a different value for the implicit parameter [23]. For example, estimates from different experts, the results of a simulation and an estimate based on previous case-histories may all contribute evidence concerning the likely value of an implicit attribute. The existence of alternative environmental models, explicit models and auxiliary models may all give rise to uncertainty in the implicit attribute values, but this uncertainty does not originate in the auxiliary model. The existence of multiple criteria for evaluation introduces variety, but not uncertainty, into the auxiliary models.

Ideally, it should be possible to represent all of the types of uncertainty identified above in the design information systems of the future. The specific aim of this research however is to support design teams in the assessment and management of risk during the design process, risk having being defined as a combination of probability and impact (on implicit design attribute values). This excludes some of the categories above from the representational requirements of the risk model. The design goals are beyond the scope of the risk model. It is not possible to evaluate probabilities from linguistically vague information and thus it is incumbent upon the users of the risk tool to resolve linguistically vague information into information concerning risk. One of the purposes of explicitly building a risk model is to avoid inconsistency - thus the model structure should preclude the possibility of inconsistencies arising and there is no requirement to represent inconsistencies. Another purpose in building the model is to assign values to constraints (such as budgets, see Chapter 4); it is therefore not necessary to represent uncertainty in such constraint values. Budgets should be regarded as point values in the context of risk management.

The types of uncertain information which the risk model should be able to represent are therefore (shown on a white background in Table 1):

- Numerical and categorical uncertainty, abstraction and incompleteness in the design environment model/s.
- Numerical and categorical uncertainty, abstraction and incompleteness in the explicit design model/s.
- Numerical uncertainty which is introduced through the use of heuristic and approximate methods in auxiliary model/s.
- Categorical uncertainty concerning alternative auxiliary models which may be used to evaluate implicit attributes.

3.3 Summary

The main aim of chapter was to identify the types of uncertain information which the risk model should be able to represent - to achieve this aim the nature of the design process which should be supported was also investigated.

In the first section, the nature of the design process was explored by conducting a review of design process models which appear in the literature. Both prescriptive models, used to advocate good design practice, and descriptive models, resulting from protocol studies where designers are observed and recorded, were presented, as well as more recent models which build on both. The models also differ in the degree of concurrency which they exhibit - with more recent models moving away from serial execution of phases and towards a more parallel approach. It was concluded in particular that designers tend to follow an evolutionary path, re-using familiar solutions to find a satisfactory solution, rather than relying heavily on synthesis, and the generation and evaluation of as many different alternative solutions as possible to find an optimal solution. It was also suggested that this emphasis on re-use may be regarded as a strength, and not a failing, as it reduces risk and uncertainty.

A key feature of many of the design process models reviewed was the concept of design as a process of *refinement* of the design model. Refinement may involve replacing abstract information with that which is more concrete, the addition of detail (addition of information), choosing between alternatives, or replacing approximate information with that which is more exact.

McMahon's view of the design process as a sequence of state transformations perfomed on *explicit*, *implicit* and *auxiliary* design models was presented and his definition of these three types of model was adopted. It was noted that there often exist multiple auxiliary models which may be used to generate a single implicit attribute with differing degrees of accuracy and differing computational costs, and that the use of an inaccurate auxiliary model will introduce uncertainty into the implicit attribute values obtained. Ward and Seering's set-based approach to concurrent engineering was also described, where many alternative design solutions are developed simultaneously and the decision between the alternatives is delayed until as late as possible in the design process. "Designers explicitly communicate and think about sets of design alternatives at both conceptual and parametric levels". This approach has the dual advantages of leading to "better" solutions than point-based design and reducing the intensity of communication which is required between team members in a concurrent engineering environment.

Current research in modelling uncertainty in early design, in order to support design decision-making, was reviewed, and a categorisation of types of uncertainty present in early design was proposed, according to the *subject* and the *nature* of the uncertainty. The categories proposed for the subject of the uncertainty were:- "design goals", "constraints", "environment of the design", "explicit models" and "auxiliary models". The categories proposed for the *nature* of the uncertainty were:- "vagueness", "numerical uncertainty", "categorical uncertainty", "incompleteness", "abstraction" and "inconsistency". Using these categories, the types of uncertain information which the risk model should be able to represent were identified and listed. It was concluded that the risk model should be able to represent numerically and categorically uncertain information, incomplete information and abstract information - all of which may concern either the explicit attributes of the artefact under design (e.g. dimensions), or its environment. The risk model should also be able to represent numerical uncertainty in the auxiliary models which are used to evaluate implicit design attributes (e.g. performance parameters), and also categorical uncertainty due to the availability of a multiplicity of such models for determining a given implicit attribute.

3.4 References

[Accar 1996] Acar, B. S., "A new model for design processes", *Proc Instn Mech Engrs*, vol. 210, no. E2, pp. 135-138, 1996.

[Allen 1992] Allen, J. K., Krishnamachari, R. S., Masetta, J., Pearce, D., Rigby, D., and Mistree, F., "Fuzzy compromise: an effective way to solve hierarchical design problems", *Structural Optimization*, vol. 4, pp. 115-120, 1992.

[Antonsson 1995] Antonsson, E. K. and Otto, K. N., "Imprecision in Engineering Design", *Transactions of the ASME Special 50th Anniversary Design Issue*, vol. 117, pp. 25-31, 1995.

[Bond 1992] Bond, A. H. and Ricci, R. J., "Cooperation in Aircraft Design", *Research in Engineering Design*, vol 4, pp 115-130, 1992.

[Boothroyd 1987] Boothroyd, G. and Dewhurst, P., "Product Design for Assembly", Wakefield RI: Boothroyd and Dewhurst Inc., 1987.

[Boothroyd 1994] Boothroyd, G., Dewhurst, P. and Knight, W., "Product Design for Manufacture and Assembly", Marcel Dekker, ISBN: 0824791762, 1994.

[Butler 1995] Butler, A. C., Coates, G. R., Rao, S. S., Sadeghi, F., and LeClair, S. R., "Modelling Uncertainty in preliminary Design : A Short Review", P*roceedings Design Engineering Technical Conferences, ASME 1995*, Boston, USA, vol. 2, pp. 371-377, 1995.

[Cross 1994] Cross, N., "*Engineering Design Methods: Strategies for Product Design*", Second Edition, Chichester, UK: John Wiley and Sons, ISBN 0 471 94228 6, 1994.

[Darke 1984] Darke, J., "The Primary Generator and the Design Process", in *Developments in Design Methodology*, ed Cross, N., Chichester : Wiley, 1984.

[Esterline 1988] Esterline, A., Rosen, D., Otto, K., Nelson, L., Hessburg, T., Riley, D., and Erdman, A., "A methodology for capturing mechanical design expertise", *Proceedings 1988 ASME International Computers in Engineering Conference*, San Francisco, August 1988, pp. 47-55, 1988.

[Finger 1989] Finger, S. and Dixon, R., "A Review of Research in Mechanical Engineering Design. Part I: Descriptive, Prescriptive and Computer-Based Models of Design Processes"., *Research in Engineering Design*, vol. 1, pp. 51-67, 1989.

[Herling 1995] Herling, D. and Ullman, D. G., "Engineering Decision Support System (EDSS)", *Proceedings of the 7th International Conference on Design Theory and Methodology*, ed Ward, A. C., held as part of the ASME Design Engineering Technical Conferences, Boston, Mass., DE-Vol 83, Vol. 2, pp. 619-626, Sept 1995.

[Hillier 1972] Hillier, B., Musgrove, J., and O'Sullivan, P.,"Knowledge and Design", Reprinted in *Developments in Design Methodology*, ed Cross, N., Chichester : Wiley, 1984.

[Hubka 1984] Hubka, V. and Eder, W. E., "*Theory of Technical Systems*", Heideleberg : Springer-Verlag Berlin, ISBN 0-387-17451-6, 1984.

[Juster 1985] Juster, N. P., "The Design Process and Design Methodologies", Technical Report, University of Leeds, May 1985.

[Konda 1992] Konda, S., Monarch, I., Sargent, P. and Subrahmanian, E.,"Shared Memory in Design: a Unifying Theme for Research and Practice", *Research in Engineering Design*, vol 4, pp 23-42, 1992.

[Locascio 1993] Locascio, A. and Thurston, D. L., "Concurrent Optimal Design with Application to Structural Dynamics", *Journal of Engineering Design*, vol. 4, no. 4, pp. 353-369, 1993.

[Luger 1993] Luger, G. F. and Stubblefied, W. A., "*Artifical Intelligence: Structures and Strategies for Complex Problem Solving*", Second Edition, Redwood City, California : The Benjamin/Cummings Publishing Company inc., ISBN 0-8053-4785-2, 1993.

[McMahon 1994], McMahon, C. A., "Observations On Modes Of Incremental Change In Design", *Journal of Engineering Design*, vol. 5, no. 2., 1994.

[McMahon 1995] McMahon, C. A., Xianyi, M., Brown, K. N. and Sims Williams, J. H., "A Parallel Multi-Attribute Transformation Model of Design", *Proceedings of the 7th ASME International Conference on Design and Methodology*, Ed. A.C.Ward, Boston, pp 341-350, Sept 1995.

[Ohsuga 1989] Uhsuga, S., "Toward Intelligent CAD Systems", *Computer-Aided Design*, vol 21, number 5, pp 317-337, June 1989.

[Otto 1993] Otto, K. N. and Antonsson, E. K., "Extensions to the Taguchi Method of Product Design", *Journal of Mechanical Design*, vol. 115, pp. 5-13, March. 1993.

[Peplinski 1996] Peplinski, J. D., Koch, P. N., Allen, J. K., and Mistree, F., "Design Using Available Assets: A Paradigm Shift in Design for Manufacture", *Concurrent Engineering: Research and Applications*, vol. 4, no. 4, pp. 317-332, December. 1996.

[Peterson 1981] Peterson, J. L., "*Petri Net Theory and Modelling of Systems*", Englewood Cliffs, NJ : Prentice Hall, 1981.

[Pahl 1988] Pahl, G. and Beitz, W., "*Engineering Design: A Systematic Approach*", London : The Design Council, ISBN 0 85072 239 X, 1988.

[Pugh 1991] Pugh, S., "*Total Design : Integrated Methods for Successful Engineering*", Wokingham, England : Adison-Wesley Publishing Company, ISBN 0-201-41639-5, 1991.

[Rowe 1977] Rowe, W. D., "*An Anatomy of Risk*", New York : John Wiley & Sons, 1977.

[Shupe 1987] Shupe, J. A., Mistree, F., and Sobieszanki-Sobieski, J., "Compromise: An effective approach for the hierarchical design of structural systems", *Computers and Structures*, vol. 26, no. 6, pp. 1027-1037, 1987.

[Simon 1969] Simon, H., "*The Sciences of the Artificial*", Cambridge Massachusetts: The MIT Press, 1969.

[Sobek 1996] Sobek, D. K., "A set-based model of design", *Mechanical Engineering*, vol. 118, no. 7, pp. 78-81, July. 1996.

[Sohlenius 1992] Sohlenius, G., "Concurrent Engineering", *Annals of the CIRP*, vol. 41 issue 2, pp. 645-655, 1992.

[Stauffer 1991] Stauffer, L. A. and Ullman, D. G., "Fundamental Processes of Mechanical Designers Based on Empirical Data", *Journal of Engineering Design*, vol. 2, issue 2, pp.113-125, 1991.

[Taguchi 1986] Taguchi, G., "*Introduction to quality engineering*". White Plains, NY: Asian Productivity Organization, 1986.

[Thurston 1991a] Thurston, D., "Design evaluation of multiple attributes under uncertainty", *International Journal of Systems Automation: Research and Applications*, vol. 1, pp. 143-159, 1991.

[Thurston 1991b] Thurston, D. L., Carnahan, V., James., and Liu, T., "Optimization of design utility", *Design Theory and Methodology*, vol. 31, pp. 173-180, 1991.

[Thurston 1994] Thurston, D. L. and Crawford, C. A., "A Method for Integrating End-User Preferences for Design Evaluation in Rule-Based Systems", *Transactions of ASME- Journal of Mechanical Design*, vol. 116, pp. 522-530, June. 1994.

[Tian 1994] Tian, Y. Q., Thurston, D. L., and Carnahan, J. V., "Incorporating End-User's Attitudes Towards Uncertainty into an Expert System", *Transactions of ASME- Journal of Mechanical Design*, vol. 116, pp. 493-500, June. 1994.

[Ullman 1987] Stauffer L.A., Ullman, D.G. and Dietterich, T.G., "Protocol Analysis of Mechanical Engineering Design", *Proceedings of International Conference on Engineering Design (ICED 87)*, Boston MA, USA, 1987.

[Ullman 1988] Ullman D.G., Dietterich T. G. and Stauffer L. A., "A Model of the Mechanical Design Process Based on Empirical Data: a Summary", AI in Engineering Conference, Palo Alto California 1988, in *Artifical Intelligence in Engineering: Design*, ed Gero, J., S., Elsevier computational mechanics publications, ISBN 1-85312-010-3, 1988.

[Ullman 1992] Ullman, D. G., "*The Mechanical Design Process*", New York : McGraw-Hill ISBN 0-07-112871-9, 1992.

[Ullman 1995] Ullman, D. G. and D'Ambrosio, B., "A Taxonomy for Classifying Engineering Decision Problems and Support Systems", *Proceedings of the 7th International Conference on Design Theory and Methodology*, ed Ward, A. C., held as part of the ASME Design Engineering Technical Conferences, Boston, Mass., DE-Vol 83, Vol. 2, pp. 627-637, Sept 1995.

[Ward 1995] Ward, A., Liker, J. K., Cristiano, J. J. and Sobek, D. K. II, "The Second Toyota Paradox: How Delaying Decisions Can Make Better Cars Faster", *Sloan Management Review*, pp 43 - 61, Spring 1995.

[Wood 1990] Wood, K. L., Antonsson, E. K. and Beck, J. L., "Representing imprecision in engineering design: comparing fuzzy and probability calculus", *Research in Engineering Design*, vol. 1, pp. 187-203, 1990.

CHAPTER 4

The Problem Domain: Management

In Chapter 4 the nature of the risk management process in a design project is explored. Support of this process is a key requirement for the risk model and modelling tool, in addition to supporting the design process and representing the uncertain design information presented in Chapter 3. In the first section, a review of project risk management methodologies is presented, with an emphasis on engineering and design projects. This indicates the types of risk which have historically been considered to be of importance to the successful completion of such projects. The second section briefly reviews the tools and techniques which may used as part of the project risk management process - mentioning established techniques and describing some recent research. An investigation has been undertaken at two collaborating industrial engineering companies into the types of uncertainty and risk which are present during the design process, and how they are perceived and managed, and the results are presented in the third section. In the fourth and final section, a summary of the requirements which were identified for the risk tool and model is presented.

4.1 Risk Management in Early Design Projects

4.1.1 Overview

Basic techniques and tools to aid the management of project risk have been in widespread use for several decades; particularly those which specifically address schedule risk such as the Programme Evaluation and Review Technique (PERT) but also more generally applicable uncertainty modelling techniques such as Monte Carlo simulation, analytic probabilistic methods (e.g. method of moments), interval-based methods and the use of decision trees, fault trees and cause-and-effect diagrams. Well developed computer based tools exist to support many of these techniques and they are described briefly in Section 4.2.2 "Risk Analysis Tools and Techniques" below. However, recent years have seen a strong emerging interest not only in new techniques and tools, but also in providing a more formal basis for the risk management methodology within which risk evaluation and analysis tools can be applied. Cooper and Chapman's "risk engineering" approach to managing large projects (Section 4.1.3 below) which was published in 1987 describes both a family of risk assessment methodologies and also a histogram-based risk modelling technique (the CIM method) in some detail. Ten years ago, risk assessment was not necessarily considered to be an integral and on-going part of managing any project, but more as a specialist task which might be undertaken at a particular moment in the lifetime of the project. More recently, however, the management of risk (including risk assessment) has started to be recognised as an important part of general project management, with a stronger emphasis being placed on the methodology and documentation and less emphasis on probabilistic modelling techniques. The RISKMAN methodology (Section 4.1.4) and the software risk management methodology developed by the

Software Engineering Institute at Carnegie Mellon (Section 4.1.6) are two recent examples which are both based on extensive project management experience. Nonetheless, most major elements of the RISKMAN methodology were present in [Cooper 1987]. In 1997 the Risk Specific Interest Group of the Association for Project Management (Risk SIG of APM) plans to publish its Project Risk Analysis and Management (PRAM) Guide; at the time of writing this is still in press, but the outline generic process has been published in [Chapman 1997] and is thus included in Section 4.1.5.

The renewed interest in project risk management can be partly attributed to a perception that there is an increased risk of time and cost overruns (with the introduction of concurrent engineering, faster time-to-market and very large scale multinational collaborative projects). There is also a rising awareness that understanding project risk is a relevant aim in itself, even where that risk cannot ultimately be removed or reduced. In particular, the Japanese model of close supplier relationships and risk-sharing necessitates an understanding of the risks which exist. The availability of the necessary software tools and computing power to make risk modelling accessible to most project managers has also clearly been important.

Much of the recent work on design project risk management has concentrated on the design of software systems [Boehm 1991] [Charette 1989] [Pressman 1992] [Ould 1990], which seems to be more inherently "risky" than other kinds of design. Huge budget and time-scale overruns, or functional failures, seem to occur more frequently in software projects than, for example, in civil engineering or automotive design. To the risks caused by the large scale and complex nature of many software design projects can be added the relative immaturity of software design as a discipline, certainly when compared with mechanical or civil engineering design for example. There is also considerable variance in productivity both between companies and between individuals within the software world and this gives rise to uncertainty in estimation of effort, costs and time-scales for software projects and hence risk. Domain-specific estimation techniques based on product and process metrics, such as function point analysis [Albrecht 1979] and Barry Boehm's construction cost model (COCOMO [Boehm 1981]), have been developed to help tackle this problem and are now widely used. More recently, causal probabilistic network modelling techniques have been applied to the problem of estimating software integrity (the probability distribution of the number of faults, a measure of product quality risk), at a given point in a development process. A network model is built of the development process and integrity is calculated using this model and many (over 100) attributes of the sub-processes and of the product [May 1993].

The defence and construction industries have also been focii for formal project risk management methods, probably due to the sheer size of their projects. There may also be high levels of technology uncertainties in defence projects. Williams has recently published a detailed review of research on project risk management [Williams 1995]; a bibliography is also available from the Risk SIG of the APM.

4.1.2 The Elements of A Risk Management Strategy

There is a broad consensus amongst writers on project risk management concerning the necessary elements of a risk management strategy; i.e. the high level activities which should be undertaken in order to manage the risk in a project. And there is agreement that project risk management is divided into phases which should be cyclically revisited during the life of the project. The naming of the phase during which an activity

should occur varies between methodologies, as does the balance between quantitative and qualitative modelling, and indeed the order in which activities should be undertaken but there is general agreement concerning the core risk management activities which should be undertaken, namely:

1. goal and base plan definition
2. risk identification
3. risk impact and probability evaluation
4. risk prioritisation
5. modelling risk relationships
6. mitigation and contingency planning
7. allocating and continually monitoring budget levels
8. continual risk monitoring

These are now discussed in more depth:

1 Goal and Base Plan Definition

Project goals generally include completing the project on time and making a profit, but there may be other goal parameters which should be identified, for example, achieving a weight budget for an automotive design project. Goals must be measurable (e.g. in units of weeks, pounds sterling or kg) but the goal value is not determined at this stage.

Also included as part of this key activity is building a model of the project "base plan". The base plan represents the anticipated structure of project elements if no risk events occur. It will generally also need to incorporate some uncertainty - for example a PERT schedule model for elapsed-time, or a product structure model which incorporates uncertainty for a performance parameter such as weight. The identified and recorded risks represent deviations from this plan. The definition and planning of the project risk management methodology is also included as part of this key activity.

2 Risk Identification

As many risks as possible are identified - uncertain events which may have a detrimental impact on the project outcome in terms of time to completion, cost or performance. Opportunities are also identified as part of this activity, where the uncertain event impact is beneficial. Each risk which is identified is assigned an "owner", who will be responsible for managing the risk/opportunity for as long as it remains a potential source of adverse/beneficial effects. The risks are recorded into some form of paper or electronic risk list or "risk register".

3 Risk Impact and Probability Evaluation

The impact and probability of each identified risk is evaluated. The evaluation may be carried out in a quantitative fashion or it may be qualitative. In a qualitative evaluation, typically, each impact is simply assigned a value of "high", "medium" or "low" and similarly for each probability. The assumption behind such a qualitative evaluation is that the identified risks are all discrete, i.e. they relate to uncertain events which will eventually occur or not occur, so there is a discrete set of only two possible impacts {0, X} where

X is the impact if the event occurs. The nature of a quantitative evaluation depends upon whether the risks identified are discrete or sliding. A sliding risk relates to an event whose impact may take any value within a range. An example of a sliding risk is "There is a risk that the printed circuit board design, which we have subcontracted out, will be late", whereas an example of a discrete risk is "There is a risk that our competitor will launch an equivalent product before our design reaches the market place". For a sliding risk, a probability distribution and its parameters must be assigned to the risk. Whereas, for a discrete risk, the impact and the probability of occurrence can each be simply assigned a numeric point value. The impacts and probabilities are recorded into the risk register.

4 Risk Prioritisation

Having evaluated the impact and probability for each identified risk, this information is used to determine which risks should be included in the risk model (usually by expected value). It is interesting and rather surprising to note that writers on project risk management generally assume that the cost of including an identified risk in the model is large in comparison to the cost of identifying it, recording it and estimating its probability and impact. Therefore it is assumed to be worthwhile to limit the size of the risk model by omitting the less significant risks. For this reason, prioritisation is listed here prior to modelling, although when the model is complete those risks which have the major effect on the overall project risk may also be identified and prioritised. In a quantitative evaluation of a complete model, a sensitivity analysis may be possible.

5 Modelling Risk Relationships

Causal relationships and interdependencies between risks are analysed and the overall impact of the identified risks on the project is thus evaluated. For a quantitative evaluation, this will involve building a risk model of some sort and may require a large analytical and/or computational effort. For a purely qualitative evaluation, the analysis is essentially subjective and takes place in the mind of the risk assessor, but much information can still be recorded as an input to the analysis. Causal relationships between risks can be recorded and the existence of interactions between pairs of risks can also be recorded using matrix notation forms. A table relating each possible <probability, impact> pair to an overall project impact can be drawn up and used to provide a qualitative measure of the overall impact of each risk on the project. The attributes modelled may be time to completion, cost or performance. Some methodologies (e.g. RISKMAN) recommend reducing all models to a single parameter, cost, at this stage. The relationships are recorded.

6 Mitigation and Contingency Planning

Those risks which have been identified as having a high priority are then controlled by a combination of mitigating their effect on the project (by changing the project "base plan" to reduce the probability of the risk, or its impact, or both) and contingency planning. A distinction is sometimes made (for example in RISKMAN) between uncertainty (e.g. in time to completion) which is caused by estimating uncertainty in the project "base plan", and uncertainty which is caused by not knowing whether or not contingency plans (e.g. requiring additional time) will need to be put into effect. It could be argued that this distinction is arbitrary - uncertainty over whether it will take one week or two to perform a task is equivalent to having a contingency plan to spend an extra week.

A single risk may well be controlled by a combination of both mitigation to remove as much as possible of its potential effect on the project, and recording a contingency plan to deal with the "residual" risk. Of course, the contingency plan may simply be to directly spend more time, to accept reduced profit or to accept reduced performance. However, where the contingency plan requires some action to be taken, a trigger event of some sort is determined and recorded, whose occurrence will indicate the need to operate the contingency plan.

The mitigating actions and contingency plans and trigger events are recorded.

Trade-off decisions may also be required. In [Klein 1993], the author discusses the concept of *intrinsic risk* in the context of risk trade-off. Project risk management is basically concerned with cost-risk, performance-risk and time-risk. Klein's view is that a certain fixed level of risk is an inherent feature of an activity, and cannot be reduced without changing the nature of the activity. This is the *intrinsic risk* and it may manifest itself as uncertainty in either time, or cost or quality. By making trade-off decisions the project manager can control how the intrinsic risk is manifested. The concept of a *load* associated with an activity is introduced (not to be confused with a design load) - a unit of activity load is the quantity of the activity that can be performed in unit time by unit personnel to standard quality. Intrinsic risk can then be measured in units of uncertainty of activity load.

7 Allocating and Continually Monitoring Budget Levels

A cost budget for the project can now be allocated at a level which explicitly takes risk into account. For a quantitative evaluation, the size of the budget will be determined by the level of profit risk which is considered acceptable and by the risk models for the residual risks and the "base plan". For a qualitative evaluation, heuristics must be used. Performance parameter budgets may also be allocated at this time - for example, weight budgets in automotive design - if they are being modelled independently of cost. The project end-date can now be predicted and this effectively defines an elapsed-time budget for the project. During the project, the margins (un-spent portion of the budgets) are continually monitored. The budget levels may also be re-allocated during the life of the project.

8 Continual Risk Monitoring

All risks which may still have an impact on the project are monitored - their probability, impact and potential overall effect on the project are periodically updated by the risk owner. Trigger events are monitored. Thus the remaining amount of cost-risk, performance-risk or time-risk in the project can be continually compared with the un-spent portions of the cost and performance parameter budgets (or the remaining time to completion date).

4.1.3 Project Risk Management Methodologies: Cooper and Chapman's "Risk Engineering" Approach

The risk engineering framework [Cooper 1987] is based on applying the basic operational research (OR) approach to a wide variety of project risk analysis problems. It is characterised by decomposition of the project into elements, and by the inclusion in the risk model of responses to uncertain events and of

dependencies between uncertain events and between project elements. When a quantitative approach is taken, the risk model is evaluated using a numerical technique based on a histogram representation for the distributions of the random variables (the Controlled Interval and Memory , or CIM , method). The problems described in [Cooper 1987] include, for example, a reliability analysis for a liquid natural gas facility in the Canadian Arctic as well as the more traditional "project risk management" task of estimating completion time for a project based on a task network which includes uncertainty. Unusually amongst writers on risk management, Cooper and Chapman's risk assessment methodologies embrace both engineering design and project management. This may be partly because of their strong emphasis on the risk model although they do also describe methodologies (risk management processes). It could be argued that, in a design project, it is impossible to completely separate the project risk model from the artefact risk model. This would suggest that the ideal of a universally applicable, domain independent, project risk management methodology or strategy may be unachievable except at a very abstract level.

> *"Very complex problems can be tackled in an effective and efficient manner only if a method is developed which is specific to the situation and the model. This method will bear some resemblance to the basic OR method but it will take advantage of a prior judgement on the nature or class of problem and the relevant class of models"* [Cooper 1987, p. 226]

In earlier work Chapman developed the Synergistic Contingency Evaluation and Review Technique (SCERT) [Chapman 1979] for time-risk analysis of North Sea offshore projects, a methodology which combines a decision-tree process representation of a project with qualitative risk assessment procedures. In [Cooper 1987] case studies are presented which illustrate other specific methodologies (based on SCERT) which have been developed within the "risk engineering" framework. The risk engineering approach involves developing methods which are problem-specific versions of the general OR methods, namely: describe the problem; formulate a model; derive a solution; test the model & evaluate the solution and; implement & maintain the solution. Figure 4-1 shows, as an example, one particular set of methods and models which could be used in a project planning situation. The family of methods described draw on a "kit-bag" of models including event-trees, decision-trees, Markov and semi-Markov processes and the CIM model (see Appendix B for details).

The key elements of the full risk engineering approach to project time planning are illustrated in Figure 4-2. Cooper and Chapman make the important point (subsequently embodied in the RISKMAN methodology) that a full, detailed quantitative analysis is not always appropriate and a choice of several levels of simplification to the full method are proposed, varying from omitting secondary risks through to omitting probabilistic modelling entirely and simply using verbal and graphical models (e.g. influence diagrams) of identified risks. The explanation below is intended as a generalisation from the specific methods described for particular cases in [Cooper 1987]. Figure 4-2 and the following explanation concern time-risk. Time risk is evaluated initially, and then converted (possibly using a probabilistic model including uncertain values for inflation, exchange rates, labour rates etc.) to a cost risk model. Other transformations may also be defined which can be used to convert impacts which are measured in other, more "natural" units (e.g. volume of earth involved in a slope failure) to cost.

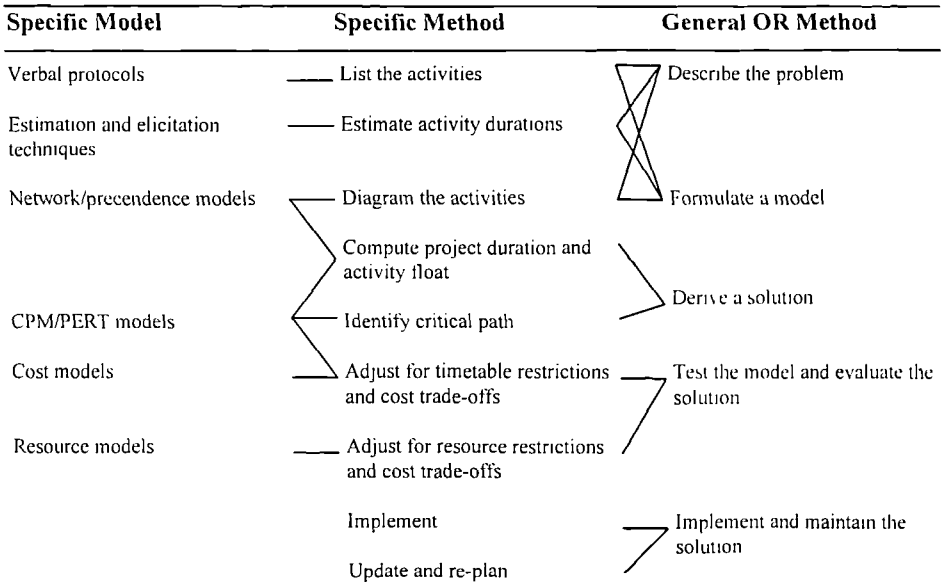


Figure 4-1: Models and methods for project planning from [Cooper 1987]

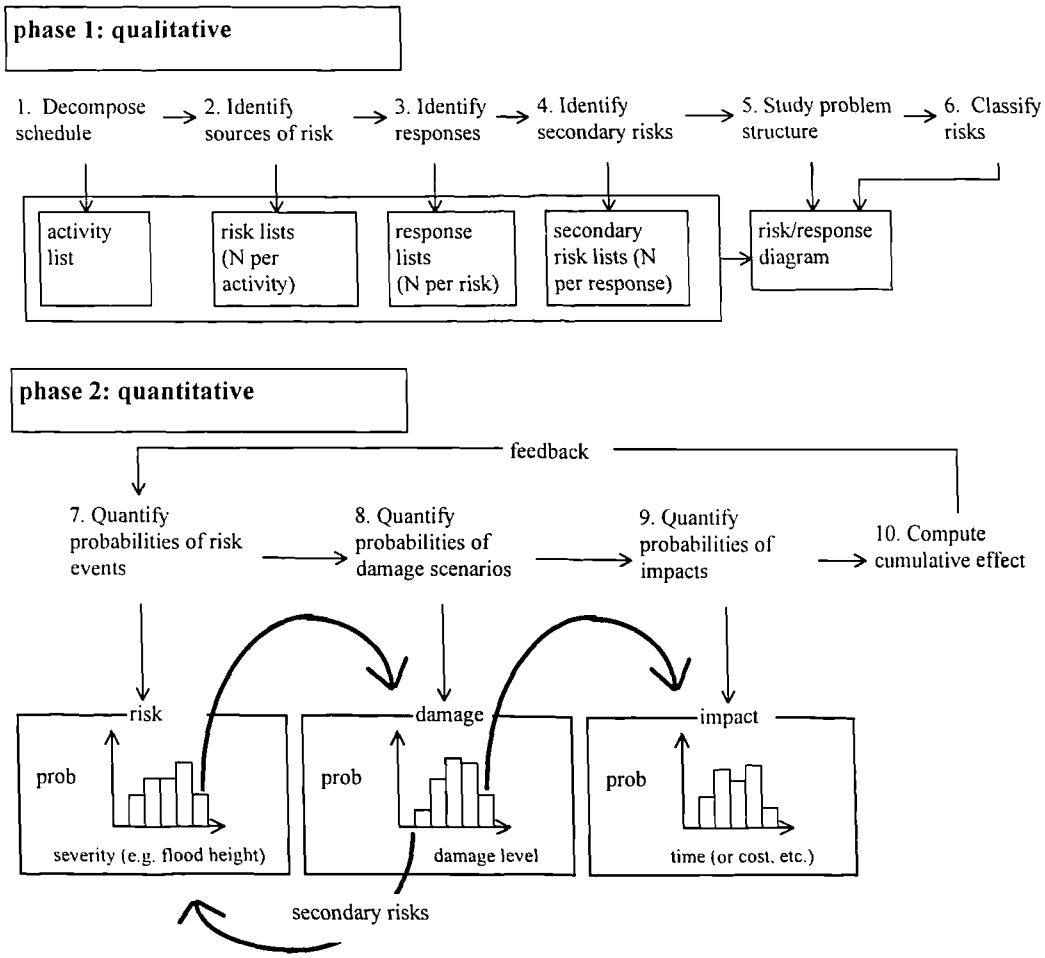


Figure 4-2: Risk engineering approach to project time-risk

The time risk evaluation is divided into two phases, the first being qualitative and the second quantitative (Figure 4-2). Initially the base (or planned) schedule is divided into a reasonably small number of activities which are listed and described. For each activity the possible sources of risk are identified and recorded; for each risk, the possible responses (mitigating actions) are identified and recorded; and where a response may give rise to secondary risks, then they are also identified and recorded. Responses which may be common to

sets of risks are identified and the decision rules which determine the choice of response are studied. A graphical network representation (risk/response diagram) of the problem structure may be generated. This is effectively a combined event and decision tree with annotations. The identified risks are classified as minor (which need not be modelled), major (which will be modelled) or beyond the scope of the analysis (the analysis is based on the assumption that these risks will not occur, thus they define the boundary of the problem). As mentioned in Section 4.1.2, above, it is surpassing that minor risks, having been identified and recorded, should then be omitted from the model - it may be that limiting the size of the model is now less important given recent advances in computing power.

The second phase, which may not be necessary in some situations, involves building a quantitative model. The probability of each risk event is assessed - for a discrete event, simply as a point probability value, but in general as a probability distribution over the severity of the event (represented as a histogram using the CIM approach). For each level of severity of each risk, the probability distribution of the possible damage scenarios is assessed, again, for full generality as a histogram - a damage scenario is a pattern of damage which may occur if a risk is realised, modelled in the general case as a probability distribution over damage level. And for each damage level the probability distribution of the impact is assessed, measured in time, or direct financial cost or in other "natural" units which can later be transformed to cost. The impact distributions are defined to include any mitigating responses identified earlier. If a response incurs a secondary risk, then this is also modelled. The CIM model also allows statistical dependencies between risks to be modelled if this is considered necessary.

Key Elements	**Risk Engineering Element**
1 goal and base plan definition	Define methodology Identify criteria (time, cost, ...) 1. Decompose schedule
2 risk identification	2. Identify sources of risk 4. Identify secondary risks
3 risk impact and probability evaluation	7. Quantify probabilities of risk events 8. Quantify probabilities of damage scenarios 9. Quantify probabilities of impacts
4 risk prioritisation	6. Classify risks
5 modelling risk relationships	5. Study problem structure 10. Compute cumulative effect
6 mitigation and contingency planning	3. Identify responses
7 allocating and continually monitoring budget levels	Use a fixed percentile of results (e.g. 90th percentile) as budget value.
8 continual risk monitoring	-

Table 4-1: Mapping of Key Elements of Risk Management Strategy into the Risk Engineering approach

This approach contains all of the key strategic elements identified in Section 4.1.2, but mitigation and contingency planning takes place prior to modelling and the responses themselves are included in the model (see Table 4-1).

Cooper and Chapman advocate the development of computer-based tools which could be used to build the risk model in parallel with cost estimates, during the design process:

> *A combined risk analysis and cost-estimating process would have a number of major benefits. A direct measure of risk, in the form of a distribution of project cost, could be obtained at the same time as the basic cost estimate was formed. This would allow any particular contingency value to be interpreted quantitatively in relation to the possible spread of project costs and probability of a cost over-run....Insight into project cost risk can be provided by enabling sources of risk to be identified and examined....The analysis provides additional information about sources of risk, and it highlights sensitive parts of the project plan and areas where design uncertainty is greatest.*
> page 169 [Cooper 1987]

4.1.4 Project Risk Management Methodologies: RISKMAN

The RISKMAN methodology, described in [Carter 1994], is the result of a Eureka research program of the same name. The RISKMAN consortium was formed in 1990 with the aim of providing improved risk management in projects, ultimately including software tools but the methodology is seen as a pre-requisite of such tools. The methodology is intended to provide a framework for risk analysis and control, rather than a detailed prescription of techniques. The aim is not necessarily to remove a risk, but to enable project participants to make an active and informed decision about which risks they wish to take and thus include a suitable contingency in the time and cost budget for the project. A software package supporting the methodology is now commercially available.

The main phases of the RISKMAN methodology are shown boxed in Figure 4-3 and map into the key elements identified in Section 4.1.2, as shown in Table 4-2.

Key Elements	RISKMAN Phase
1 goal and base plan definition	
2 risk identification	identification
3 risk impact and probability evaluation	assessment
4 risk prioritisation	evaluation
5 modelling risk relationships	evaluation
6 mitigation and contingency planning	mitigation
7 allocating and continually monitoring budget levels	contingency estimation decision making
8 continual risk monitoring	control and monitoring

Table 4-2: Mapping of Key Elements of Risk Management Strategy into RISKMAN

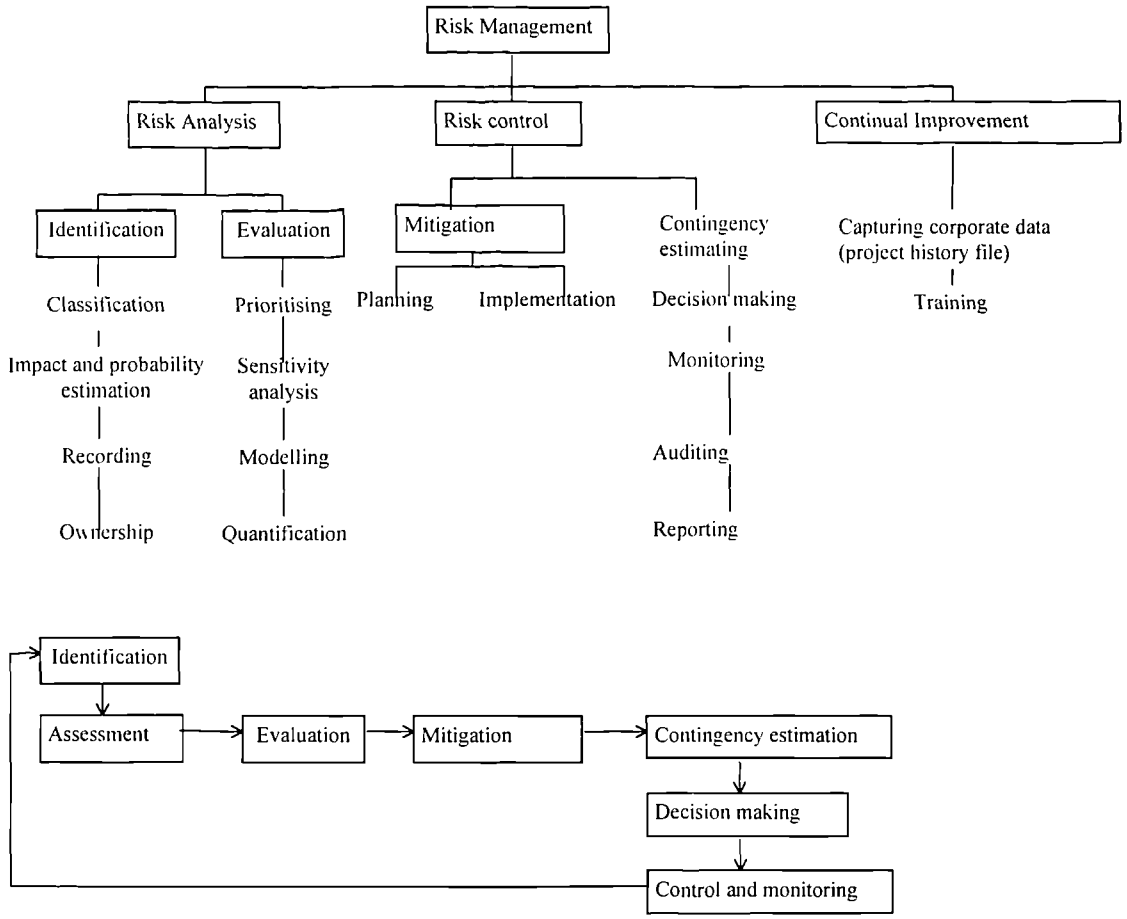


Figure 4-3: The RISKMAN project risk management model, from [Carter 1994, p. 73]

Although RISKMAN incorporates much of the risk engineering approach to schedule risk and project cost risk, there are significant differences. RISKMAN emphasises the cyclic, iterative nature of the risk management process and also concentrates more on the documentation, monitoring and control aspects (examples of form layouts for recording information, risk ownership, risk lifetime, etc.) and less on the modelling aspects than Cooper and Chapman. This is partly attributable to the shift in emphasis in recent years from risk analysis as a specialist activity to risk management as a necessary component of all project management. Cooper and Chapman's risk model is capable of considerably more sophistication than the models described in [Carter 1994]. RISKMAN advocates that mitigation (changes to the base plan to incorporate responses to risk) is undertaken after evaluation of the risk model, whereas a distinguishing feature of Cooper and Chapman's approach is the incorporation of these responses into the risk model. Some of the key concepts from RISKMAN are briefly explained below:

Key Concepts

risk, risk ownership and risk typology

The definition given is that a risk "involves uncertainty and has an impact" ([Carter 1994, p. 16]), where the impact may be either beneficial or detrimental to the project. The phrase "involves uncertainty" is interpreted

here as meaning "involves an uncertain event", and in what follows this event is referred to as the risk event. All risks have a cause and an impact, and all risks are assigned an owner.

Three categories of risk are identified, where the category of a risk indicates the extent (but not necessarily the magnitude) of the effects if the risk event occurs. Category 1 risks involve uncertainty concerning how long something will take or how much it will cost. If a category 2 risk event occurs then contingency actions must be taken at a local level but the "main plan" is unaffected. If a category 3 risk event occurs then the main plan must be changed.

Twelve classes of risk are also identified, where the class of a risk indicates the area of impact of the risk event - for example, the risk that the sales volume achieved will be less than anticipated would fall into the "marketing" class. These twelve classes are then grouped into four baselines, where each baseline represents a part of the project lifecycle: strategic, definition, technical and operation. It is suggested that a company should add its own specialised classes to reflect its business area.

impact of risk

An impact may be measured in units of cost, time or quality. All impacts are listed and each is assigned a single risk class. If a single risk has impacts in more than one class then it must be decomposed into units which lie within a single class. This is necessary because the sets of potential owners for each risk class are generally disjoint and it is required that a single owner should be identified for each risk.

causes of risks

One risk may have many causes, but, once again, each individual cause must lie within a single class, for reasons of owner assignment. A cause may be a constraint (e.g. the project must be completed by a certain date), another risk or a fact (e.g. the technology to be used is immature). Thus a causal network of risks is defined. It is a rule of the RISKMAN methodology that every risk without direct financial impact must lead via the causal network to one or more risks with financial impact.

risk analysis level

The risk analysis may be performed at three levels. In a basic level analysis, the probability of a risk event occurring is assigned a qualitative value, for example "high"/"medium"/"low", as is its impact. In an intermediate level analysis, the probability of a risk event occurring is given as a percentage figure with a point value for the impact. The comprehensive level of analysis requires specification of the probability of a risk event as a PDF and its parameters (for example, a normal distribution with mean of 10.00 and standard deviation of 3.6). In RISKMAN, unlike Cooper and Chapman's approach, the basic level of analysis is not truly qualitative since the linguistic value is subsequently interpreted as a numerical value and used as input to a numerical model for risk aggregation.

constraints

For those constraints which appear in the risk causal network, details of "target" and "real" values are recorded, both of which may be uncertain. The extent to which the target and real values can be modified is recorded as is the degree of "dispersion" between them and the cost of failure to meet the constraint. No definition of dispersion is given for a comprehensive or intermediate level of analysis, but the probability of achieving the target value would seem to be a reasonable measure.

trigger events

A trigger event can be recorded against each risk. This is the time event at which the planned risk mitigation activity will be initiated.

active period

"...the time period during which a risk may occur, starting with the event that triggers its inception right through to the end of its effects on the programme or activities." [Carter 1994, p. 51]

generation period

"...the period during which a factor or combination of factors may lead to its [the risk's] existence." [Carter 1994, p.51]

risk budget

As mentioned above, the RISKMAN methodology does not prescribe techniques to be used for risk analysis - a choice of qualitative or quantitative techniques are suggested (termed basic, intermediate or comprehensive application levels), any of which may in principle be used within the framework. However, the method does prescribe rather precisely how a suitable level for the project budget should be determined, at the intermediate or comprehensive application levels, so as to explicitly take risk and uncertainty into account (see [Carter 1994, pp. 75-78]).

The project budget is decomposed into the six prime cost elements shown in Figure 4-4. Two separate PDFs are built up - the estimation process generates a *base cost curve* which represents the PDF of the cost of the "base plan"; and the other processes shown generate a *risk cost curve* which represents the PDF of the accumulated cost of all the identified risks (uncertain variations on the base plan) which have not been removed by mitigation activities. The high probability risks (e.g. above 50% likelihood) are also removed and listed separately, before constructing the risk cost curve.

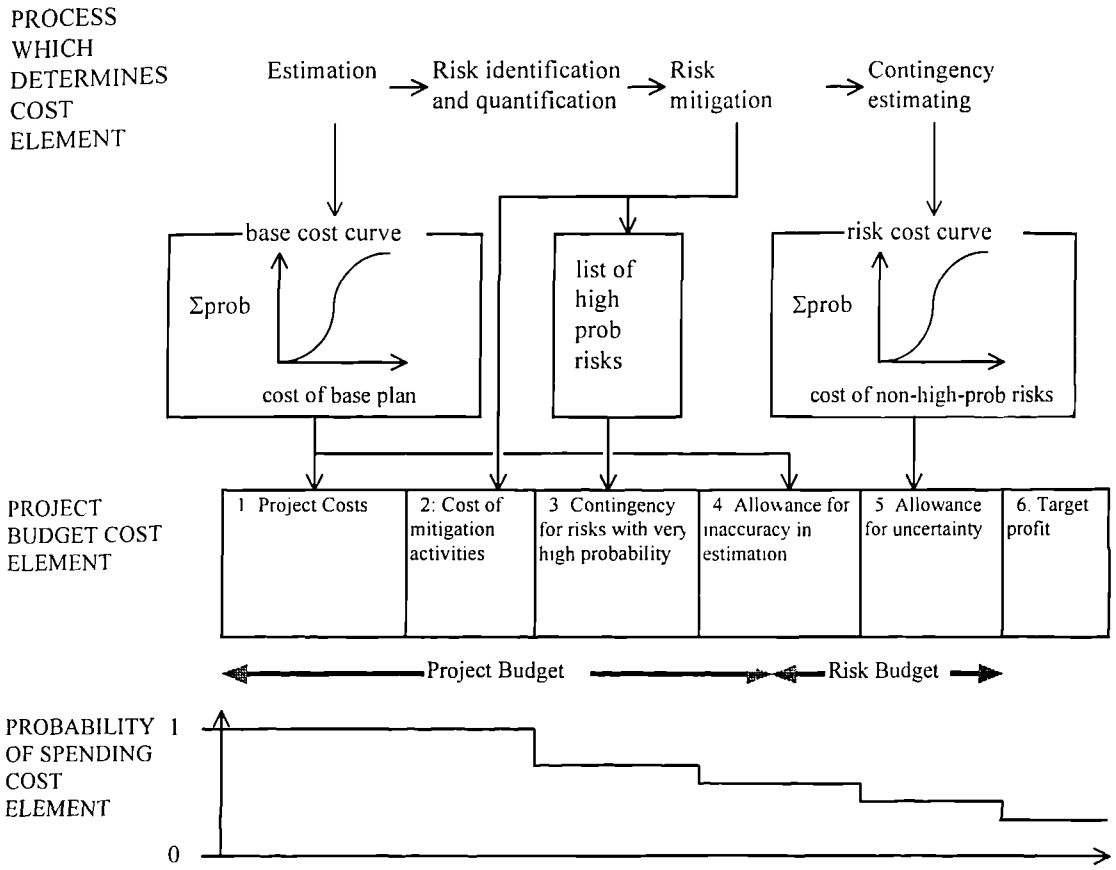


Figure 4-4: RISKMAN's cost elements of project budget, from pages 75-76 [Carter 1994]

The levels of cost elements 1, 3, 4 and 5 are then determined using a simple algorithm from the two cost curves and the list of high probability identified risks (each with an evaluated probability and cost impact), as follows:

Element 1 (Project costs): take the 50th percentile of the base cost curve

Element 3 (Contingency for risks with high probability): take the whole of the sum of the impacts of all the listed risks whose probability exceeds 50%

Element 4 (Allowance for inaccuracy in estimation): take the (70th percentile - 50th percentile) of the base cost curve

Element 5 (Allowance for uncertainty):take the 60th percentile of the risk cost curve.

The values of 50%, 60% and 70% are suggested example percentages, not hard and fast rules. Once the budget levels have been determined using this algorithm, the residual "potential for loss" is calculated, which is the maximum amount of the profit which may be consumed during the project, according to the current information, if these budget levels are adopted. Using the example percentages above, it will be given by:

potential for loss = (maximum - 70th percentile) from base cost curve +
(maximum - 50th percentile) from risk cost curve

There are two criticisms which could be made of this algorithm. Firstly, it is based on an unrealistically simple model. All deviations from the base plan must be quantified as discrete (not sliding) risks and they must be independent. Secondly, and more importantly, it does not yield results which are meaningful in terms of probability. This is because of the way in which cost element 3 is calculated - the probability that all the (independent) risks with above 50% likelihood will be realised is the product of their individual probabilities and may be large or small depending on the distribution of the high probability risks. The decision maker must judge the various percentage levels to be used in the algorithm (or they may be defined as part of a company policy), but they cannot be interpreted as probabilities and do not determine the probability that the project will be completed within budget. It could be argued that such heuristic "recipes", which have evolved as a product of management experience but which are not consistent with the rules of probability, do not engender good risk estimation skills. The reason for the inclusion of the full impact of all high probability risks is presumably to counter-effect the well-known tendency towards optimism in risk estimation, but by hiding the effect of this optimism, the possibility of feedback from experience on many projects increasing accuracy in estimating probabilities is reduced.

These type of difficulties often arise from attempts to avoid or "simplify" numerical models by combining qualitative and quantitative approaches. It is suggested, however, that adding the "base costs" and the "risk costs" to obtain a single PDF and then taking a specified percentile as the budget value is in fact much simpler than the approach described above. As suggested earlier in this section, the distinction between "base costs" and "risk costs" is essentially arbitrary from a modelling perspective, although relevant from a project planning perspective.

4.1.5 Project Risk Management Methodologies: The Association for Project Management's PRAM Guide

As mentioned above, the PRAM guide has not yet been published, but it will include the "generic risk management process structure" given in [Chapman 1997]. The nine phases shown in Table 4-3 and in Figure 4-5 are identified - they are visited iteratively and cyclically as indicated in the figure, though for clarity, only the particularly important feedback loops are illustrated.

Key Elements	PRAM Guide Phase
1 goal and base plan definition	define, focus
2 risk identification	identify (risks), ownership
3 risk impact and probability evaluation	estimate
4 risk prioritisation	evaluate
5 modelling risk relationships	structure
6 mitigation and contingency planning	plan, identify (responses)
7 allocating and continually monitoring budget levels	plan, manage
8 continual risk monitoring	manage

Table 4-3: Mapping of Key Elements of Risk Management Strategy onto the PRAM Guide Phases

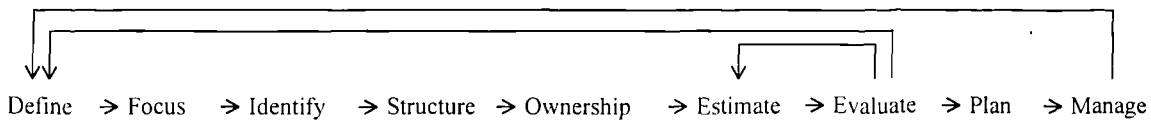


Figure 4-5: Chapman and Ward's Generic Risk Management Process, from [Chapman 1997]

The *define* and *focus* phases involve identifying project objectives and scope, defining high-level activity base plans and scoping and planning the risk management methodology itself. During the *identify* phase, risks and responses are identified, as are secondary risks (arising from responses). The *structure* phase involves analysing the relationships between risks, responses and activities - it may involve building mathematical models. During the *ownership* phase, the extent to which risk will be shared with sub-contractors is determined, as well as individual owners being allocated to those risks which are the responsibility of the major company. *Estimation* involves estimating likelihood and impact of identified risks, numerically or qualitatively. This information is then *evaluated* to yield an indication of overall level of risk and of the most important contributors to the overall risk. The *plan* phase involves building a detailed activity base plan including expenditure and a risk management plan including contingency plans with trigger points. Once the project is implemented, the risk must be *managed* - progress is monitored, plans adjusted or re-formulated in the light of events, trigger events are responded to.

4.1.6 Project Risk Management Methodologies: The SIE's Software Risk Evaluation Method

The Software Engineering Institute at Carnegie Mellon University have developed a qualitative software risk management methodology which they term the Software Risk Evaluation (SRE) method [Sisti 1994]. The SRE method is based on a knowledge engineering approach and a *risk taxonomy* (see Section 4.2.1) has been developed which is central to the method. The main activities of the SRE methodology, which are executed cyclically, are shown in Figure 4-6 and map into the key elements identified in Section 4.1.2 as shown in Table 4-4.

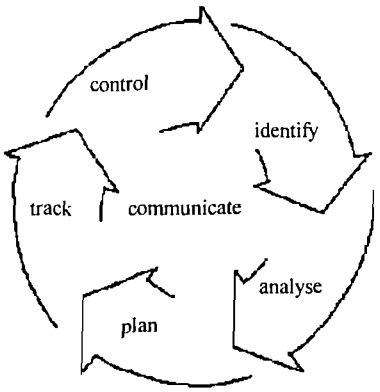


Figure 4-6: Carnegie Mellon's SEI software risk management model [Sisti 1994]

Key Elements	SRE Phase
1 goal and base plan definition	
2 risk identification	identify
3 risk impact and probability evaluation	analyse
4 risk prioritisation	analyse
5 modelling risk relationships	analyse
6 mitigation and contingency planning	plan
7 allocating and continually monitoring budget levels	track
8 continual risk monitoring	track

Table 4-4: Mapping of Key Elements of Risk Management Strategy into the SRE Phases

In the *identification* activity, a taxonomy-based questionnaire (TBQ) is used to elicit as many risks as possible from the project team (described in Section 4.2.1 "Risk Identification Tools and Techniques" below). The *analysis* activity is defined as the conversion of risk data into decision-making information. Effectively, this consists of building and evaluating a risk model. *Planning* is defined as the conversion of decision-making information into plans and actions; this includes planning both mitigating actions and also the acquisition of further information concerning a risk, where more information is needed to inform subsequent decisions. During *tracking*, suitable metrics of overall project risk are identified and monitored, trigger events are identified and mitigating actions are monitored. *Control* consists of correcting for deviations from planned actions; this may involve key elements 1 through 7. An additional activity, *communication*, is identified and is seen as central to all the other activities.

The primary functional components of the SRE are detection, specification, assessment, consolidation and mitigation. Risks are *detected* using the TBQ. The *specification* of a risk includes its conditions (identifying the circumstances under which the risk may occur), its consequences and its immediate source, and may be expressed in informal natural language or represented more formally using a structured syntax similar to the following:

given <condition> **then (possibly)** <consequence>

Risk *assessment* consists of associating a qualitative "magnitude" (or exposure level) with each risk, which is one of "critical", "high", "medium" or "low" and represents the expected value of the risk, i.e. the product of probability and impact. During *consolidation*, information obtained from multiple interview sessions or multiple evaluations on a single project is combined. Instances of multiple descriptions of a single risk are identified and any differences or inconsistencies between different parts of the risk information are reconciled. *Mitigation* is facilitated by grouping similar risks into "risk mitigation areas". For each risk mitigation area, the current status is analysed and recorded, the desired status is recorded (as specific goals which support the project goals) and mitigation strategies and activities are then developed and recorded. A risk map may be drawn up relating risks to mitigation areas and also relating risks to project goals.

The SEI are currently developing a predictive decision model for software risks; the aim being, in the absence of good risk data about a new project, to be able to use historical data about previous similar projects to help predict the risks and mitigation actions for the current new project. The predictive decision model is intended to be an abstract risk model which is applicable to any software project. The results of any SRE method application are therefore stored into a database, once all project or company identifying information has been removed.

4.2 Tools and Techniques

Having reviewed project risk management methodologies in the previous section, we now briefly describe some of the tools and techniques which may be used to implement these methodologies. This section is divided into tools and techniques for risk identification, and those for risk analysis.

4.2.1 Risk Identification Tools and Techniques

The techniques used to identify risks as part of the project risk management process range from simply applying common sense and experience through to formal "risk review" procedures, where an independent expert questions project team members in detail about every aspect of the project which might give rise to risk. Questionnaires, interviews and checklists are all used in an attempt to elicit as much relevant information as possible from the project team members and also to provoke thought concerning likely areas of risk.

Historical information may be used informally, as part of the risk assessor's knowledge and experience or more formally, as in the SEIs taxonomy-based questionnaire [Carr 1993]. The assumption behind the taxonomy-based questionnaire (TBQ) is that there is much information concerning project risks which is present in the minds of the project team members but is not, generally, gathered, made explicit and acted upon. The SEI have developed the software taxonomy shown in Figure 4-7. The taxonomy is based on information from the literature on software development and risks (e.g. [Boehm 1991]), as well as SEI team members' experience and also analysis of the results from field trials of the method. It identifies characteristics of software development and hence, it is assumed, of software development risks. A questionnaire has been developed which presents a structured set of questions concerning every attribute in the taxonomy. This questionnaire is used as the starting point for interviews conducted by independent assessors with small groups of project staff. One interesting finding from the SEI field trials is that it is essential to include only peers in an interview group - the presence of a manager is invariably inhibiting.

CLASSES	ELEMENTS	ATTRIBUTES
Product engineering	Requirements	Stability Completeness Clarity Validity Feasibility Precedent Scale
	Design	Functionality Difficulty Interfaces Performance Testability Hardware Constraints Non-Developmental Software
	Code and unit test	Feasibility Testing Coding Implement-ation
	Integration and test	Environment Product System
	Engineering specialities	Maintainability Reliability Safety Security Human Factors
Development environment	Development process	Formality Suitability Process Control Familiarity Product Control
	Development system	Capacity Suitability Usability Familiarity Reliability System Support Deliverability
	Management process	Planning Project Organization Management Experience Program Interfaces
	Management methods	Monitoring Personnel Management Quality Assurance Configuration Management
	Work environment	Quality Attitude Cooperation Communication Morale
Program constraints	Resources	Schedule Staff Budget Facilities
	Contract	Type of Contract Restrictions Dependencies
	Program interfaces	Customer Associate Contractors Subcontractors Prime Contractor Corporate Management Vendors Politics

Figure 4-7: The SEI software development risk taxonomy

The RISKMAN methodology includes a similar but far more general taxonomy, shown in Figure 4-8, which is intended to be applicable to projects in all domains. It serves both as a classification to organise identified risks and indicate who would be the most appropriate owner but also, like the SEI taxonomy, as the basis of a check list, to provoke thought around issues which have historically proven to be risk related.

CONFIGURATION	RISK CLASS	RISK CHARACTERISATION
Strategic baseline	1.Strategic	Quality of the strategic plan. Likelihood of failure to achieve plan.
	2.Marketing	Quality of requirements definition. Likelihood of failure to achieve marketing plan. Commercial relations.
Definition baseline	3.Contractual	Legal risks.
	4.Financial	Financial risks.
	5.Master plan	Likelihood of failure to meet major milestones and costs.
	6.Definition	Likelihood of failure to meet product requirements.
Technical baseline	7.Process (work breakdown structure)	Implementing a particular process.
	8.Product (product breakdown structure)	Developing a particular architecture.
	9.Organisation (organisational breakdown structure)	Managing the organisation.
Operation baseline	10.Operational	Harm or injury caused by product operation.
	11.Maintenance	Cost of maintenance.
	12.External	Problems created by, or for, the environment or socio-political world.

Figure 4-8: The RISKMAN risk taxonomy

The idea of building a knowledge base for risk identification is not new. Niwa [Niwa 1989] built a knowledge-based computer system (an expert system) to provide a project manager with knowledge about risk gathered from many project managers. His "human-computer co-operative system" uses a model of risk causes to warn managers of risks which may follow from causes which they have specified and also to assess the likelihood of hypothesised risks. Knowledge relating to many hundreds of risks from historical projects was collected and represented within a structured production system, i.e. as production rules of the form:

if <condition> **then** <consequence>

Both backward and forward reasoning were implemented as the inference methods, but a "knowledge association" method was also implemented, based on keyword retrieval, to retrieve non-logically-related knowledge. The aim here was to incorporate human intuitive processes into a traditional expert system which can already emulate the usual logical thought processes.

Niwa's work showed that similar risks did indeed recur across projects within the particular domain with which he was concerned (large scale construction projects - the case study presented is for a thermal power station). In a survey which Niwa quotes [Niwa 1981], 3 out of 8 companies said that 80-100% of the risks they encountered were familiar, and only 1 company said that less than 50% were familiar.

The basic model of risk causes which Niwa used is shown in Figure 4-9: risk factors give rise to risks, which may then give rise to other risks.

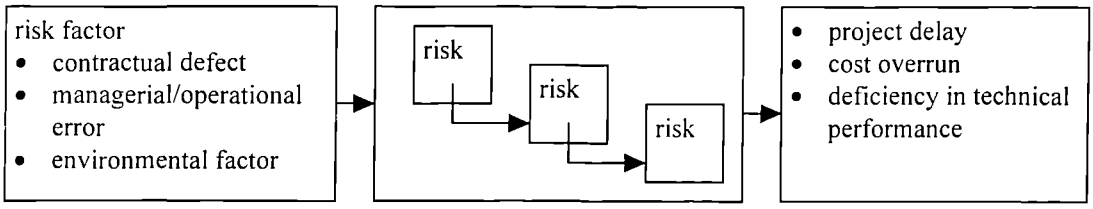


Figure 4-9: Niwa's causal risk model

A risk gives rise to schedule delay, cost overspend or reduction in technical performance. Three classes of risk factor were identified - contractual, managerial/operational and environmental. Comparing these with the RISKMAN taxonomy, note that Niwa's work does not concern the pre-contract phase and hence the managerial/operation class encompasses RISKMAN classes 4, 5, 6, 9, 10 and 11. RISKMAN classes 7 and 8 are dealt with rather differently using a "standard work package method" illustrated in Figure 4-10.

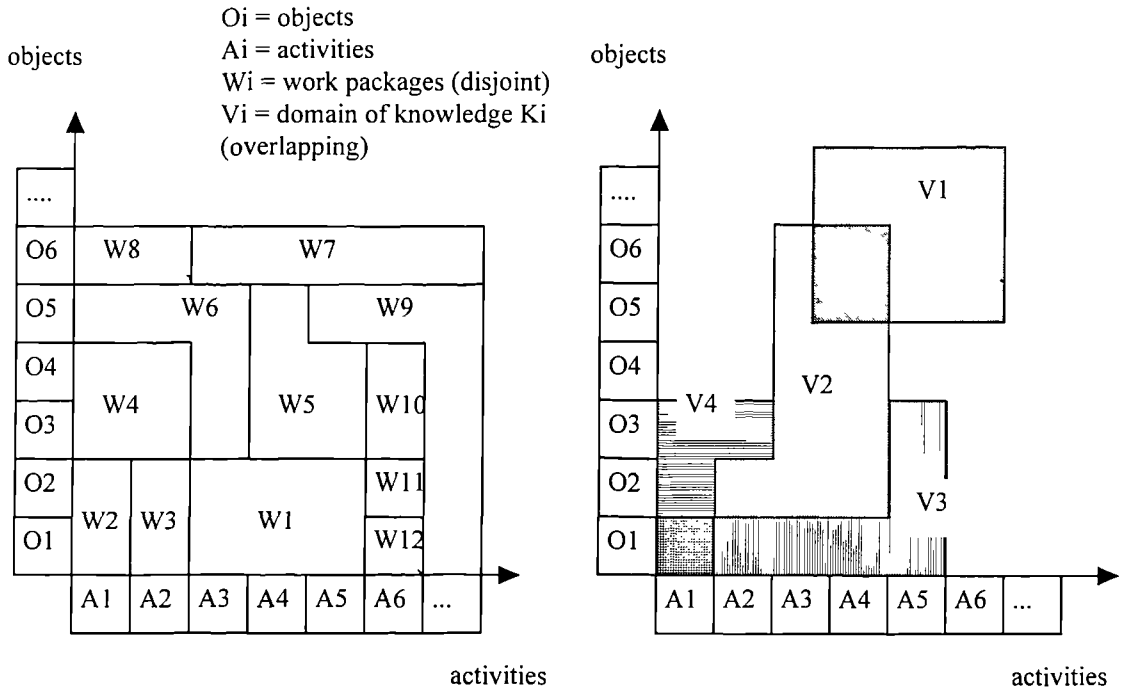


Figure 4-10: Niwa's standard work package method

A work package is defined in a two-dimensional space which has a set of standard activities along one axis (e.g. procurement, installation, etc.) and standard objects (equipment or buildings, parts of the product model) along the other. The standard activities and objects are chosen to be applicable to any large scale construction project. The domain of a piece of knowledge is defined in the work-package space and so are

the work packages in the project of interest. Thus by superimposing the work package map on the knowledge domain map and finding the overlaps, one may identify the pieces of knowledge which are relevant to a given work package and, conversely, the work packages which are relevant to a given piece of knowledge.

Managers were asked to provide knowledge in the form of a risk, possibly consequent risks, its causal risk factors and the activities and objects to which it applied (thus defining the work package which is its domain). This knowledge was then represented in production rules such as:

if <risk factors> **then** <risk factors>
if <risk factors> **and** <work packages> **then** <risks>
if <risks> **and** <work packages> **then** <risks>

and backwards and forwards reasoning methods implemented. This was augmented by a "knowledge association" method for building dynamic associations, which are not part of the knowledge base, between a starting risk and other risks. The human user provides keywords of interest and the computer system returns examples of risks which include that keyword. The keywords are not defined in advance, thus it is basically a free-text search, but the computer system provides suggestions, tailored to the position, purpose and skill level of the user, to guide the association. The results of the knowledge association method can optionally be incorporated into the knowledge base as new rules .

It could be argued that Niwa's need to include the informal knowledge association method illustrates the fundamental difficulty of implementing expert systems for risk identification; namely the ill-structured nature of the problem domain. But, his findings that the same risks do recur repeatedly indicates that some structure exists and hence such tools may be of considerable benefit.

4.2.2 Risk Analysis Tools and Techniques

Many general probabilistic analysis techniques are currently applied to project risk management, with the aid of suitable computer-based tools. Appendix B contains an overview of Monte Carlo simulation, event trees, decision trees and numerical methods such as numerical convolution, discrete probability distributions and controlled interval and memory.

Schedule-specific techniques such as PERT and critical path analysis [Moder 1970] are well-known and well-supported by currently available commercial software.

Rao and Putcha have used Rosenblueth's two-point estimate method [Rosenblueth 1981] to evaluate the probability distribution for project completion time based on a task network with precedence relations [Putcha 1991]. In [Rao 1993], the application of utility theory [Keeney 1976] [Eeckhoudt 1995] to engineering project choices is illustrated. Utility theory provides a means to quantify the risk aversion (or conversely the risk proneness) of a decision maker, for example as the coefficient of an exponential utility function, and thus evaluate the utility of alternatives which are quantified as probability distributions rather than point values. Rao calculates the distribution of the present-worth cost of each alternative under consideration using Monte Carlo simulation and also using Rosenbleuth's two-point estimate method. The

"net present utility" of each alternative is assessed, showing that the "best" alternative is not necessarily the one which has the lowest expected cost value. Borrowing from reliability theory, the safety index β of the decision is also evaluated, providing a way to determine which alternative has the lowest probability of being the "wrong" choice.

In [Smith 1995] a technique is proposed for choosing the optimum ordering of a set of project tasks to minimise the expected value of the cost. Each task is allocated a cost plus a probability that the project will fail as a result of that task. Precedence relationships and parallel tasks can be modelled but it is assumed that the probability of failure and the cost are independent of everything except the task identity and also that the cost is known precisely for each task. The use of expected value implies risk neutrality and the algorithm assumes no rework - each task is only executed once.

Many companies have developed their own tools to support their in-house project risk management method. For small projects, using a qualitative risk model, a paper-based system of forms may be adequate. For larger projects, simple custom database applications which store and index qualitative information in a simple risk register and generate summary reports are often used. Commercial software is available which supports the maintenance of a risk register dynamically linked to a quantitative risk model during the project life cycle; Monte Carlo simulation is used to evaluate the resultant schedule, cost and resource distributions.

As part of Esprit project 6283, City University have developed a software tool called GOAL [Cowderoy 1995] which supports a risk management process, outlined below, which is broadly consistent with RISKMAN. The tool is being developed as an add-in to existing third-party spreadsheet and Monte Carlo software. The main stages in the GOAL process are shown below:

1. *goal definition*: Defining measurable objectives
2. *event management*: Building a risk register of sliding or discrete risks with their associated probabilities and impacts. Storing "control actions" (mitigating actions or collecting further data on a risk).
3. *cashflow analysis*: Analysing the timing of income and of risk events, to highlight where expenditure may exceed income.
4. *work planning*: Building the work breakdown structure for the project, allocating resources, performing critical path analysis etc. Identifying risks and opportunities associated with particular tasks.
5. *work output planning*: Building the product breakdown structure for the output of the project, associating deliverables with tasks. Identifying risks and opportunities associated with particular deliverables.
6. *co-ordination*: Allocating owners to identified risks, communicating changes in the risk model to the appropriate people.
7. *risk monitoring*: Watching changes over time in important quantities which impact on risk.

A recent development in knowledge-based risk analysis is the work at the Software Engineering Institute at Carnegie Mellon on knowledge summarisation, analysis, and visualisation (K-SAV) technology [Monarch

1995]. Using a technique called co-word analysis, Monarch and co-workers attempt to extract the most important risks and their relationships from the textual descriptions of the identified software project risks which were obtained using the taxonomy based questionnaire interview method described above. The output from the co-word analysis is represented by terminological networks termed "lexe-mappes" which show the relationships between concepts which were present both implicitly and explicitly in the analysed text. The concepts are identified by looking for sets of phrases which all share a common term; the shared term is a concept. The strength (represented by line weight) of an arc relating two concepts in a terminological network then represents how often the concepts occur together compared to how often they occur separately. Initial experimental results have been encouraging; the sets of most important risks identified using the technique covered most of those identified by interviewing human experts, the relationships exhibited were largely explicable and, importantly, included some relationships which were not explicitly present in the risk statements. Monarch and co-workers emphasise that co-word analysis and lexe-mappes are more than simply automated analysis tools; they offer new modes of communication which increase opportunities for knowledge sharing and can thus be viewed as media for the representation, communication, and integration of knowledge into management processes.

4.2.3 Summary

In Section 4.1, the eight key elements of a project risk management strategy were identified, and risk management methodologies from the literature were reviewed. In Section 4.2, the tools and techniques for project risk management were explored. There are a wide variety of techniques and computer-based tools available to support conventional probabilistic risk analysis for project risk management; it seems plausible that the reason that they are not in more widespread use in engineering design projects is not lack of availability but the considerable investment of additional time required during the project to use such tools effectively. There is a need for new approaches which expedite the generation of risk models. The development of knowledge-bases of taxonomies and checklists may help serve this purpose - both Niwa and the Software Engineering Institute at Carnegie Mellon find that risks recur.

4.3 Current Practice of Collaborating Industrial Partners

In Sections 4.1 and 4.2, several theories of project risk management were presented along with tools and techniques for their implementation - with an emphasis on engineering and design projects. In this section, we describe risk management practice for design projects at two industrial engineering companies - Cegelec Projects Ltd and Rover Group Ltd. The types of risk which are present are described, and the ways in which risk is perceived and managed. The aim of this section is to compare theory with practice.

4.3.1 Perceptions of Risk at Cegelec

Definition of Product and Early Design Phase

This assessment is based on the case-study presented in [Watson 1994]. The case study concerns a project to design a tandem cold rolling mill which is a process area in a steel works. This is in fact only part of a larger project, but for the purposes of risk assessment can be regarded as representative of a whole project. The early design process consists of construction of a tender document, containing a commitment to meet the

client's requirements for a quoted price. A functional breakdown is used in the preparation of the tender document. The top three layers of the functional breakdown in the case study are shown in Figure 4-11.

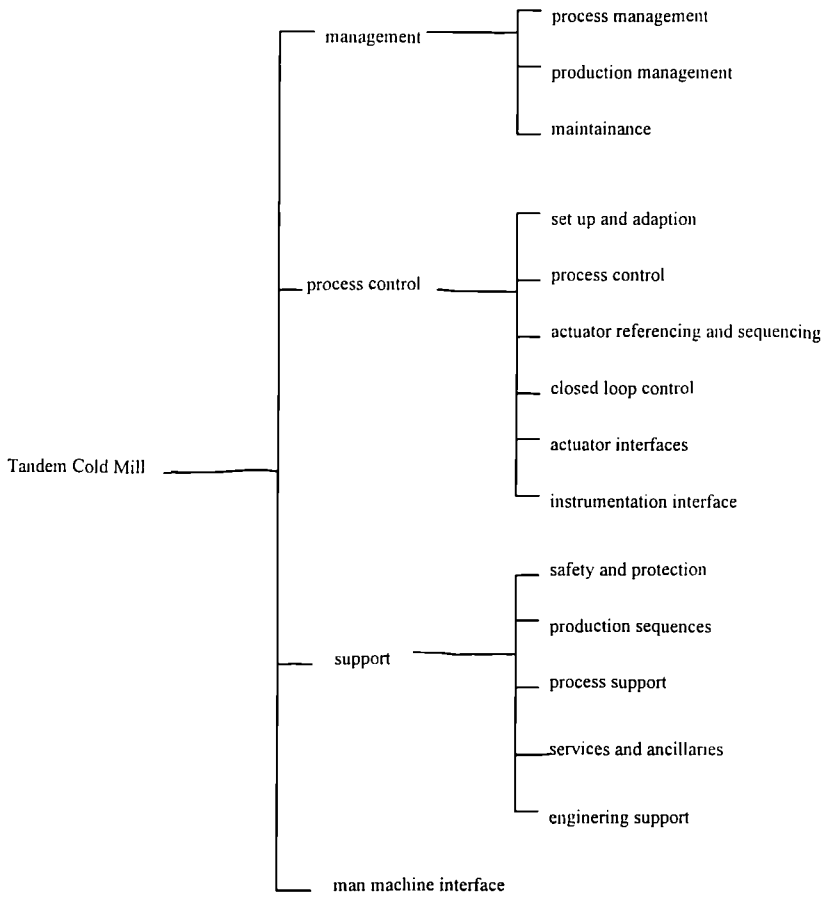


Figure 4-11: Functional breakdown for tender document

During the development of the tender, each function in the hierarchy has four costs associated with it:

AA_I = the cost of supplying the function "as is", i.e. without any further development work

Devp = the cost of performing some proposed development work

Dev_A = the cost of that part of the proposed development work which has already been authorised (and so is not part of the costs for this contract)

AA_D = the cost of supplying the function after the proposed development work has been completed

These costs are all generally measured in units of "man weeks of engineering effort". These costs are then rolled-up over the project to produce an overall estimate of $Cost_{asis}$, $Cost_{afterdev}$ and $Cost_{contract}$. $Cost_{asis}$ is the cost to supply the project without any further development work, $Cost_{afterdev}$ is the cost to supply the project after the proposed development work has been completed and $Cost_{contract}$ is the total cost of fulfilling the contract if the development work is to be completed as part of the contract:

$$Cost_{asis} = \sum_{\text{all functions}} AA_I + \text{integration costs}$$

$$Cost_{afterdev} = \sum_{\text{all functions}} AA_D + \text{integration costs}$$

$$Cost_{contract} = Cost_{afterdev} + \sum_{\text{all functions}} [Dev_P - Dev_A]$$

Risk Identification

$Cost_{asis}$ is used as a basis for the price quoted in the tendering document. The identified project risk is the risk that $Cost_{asis}$ is under-estimated, resulting in a small (or even negative) difference between the quoted price and the actual price to supply. Conversely, the possibility that the difference between the quoted price and the cost to supply is larger than anticipated, due to an over-estimate of $Cost_{asis}$, is identified as an opportunity. It is implicit in the means of analysis (below) that the impact of an under-estimate on the project is assumed to be equal to the amount of the under-estimate and similarly for an over-estimate.

Risk Analysis

Each function has a *development status* assigned to it in addition to the four costs given above. This is an integer from 1 to 8 which represents the degree of uncertainty in AA_I. It is interpreted as indicating the maximum and minimum possible cost-to-apply as a percentage of the estimated value, as shown in Table 4-5.

Development Status	Lower limit, as percentage of AA_I	Upper limit, as percentage of AA_I
1	50%	200%
2	50%	150%
3	50%	150%
4	75%	125%
5	85%	115%
6	85%	115%
7	90%	110%
8	95%	105%

Table 4-5: Percentage Limits According to Development Status (Cegelec Projects)

The impact of these risks on the project is assessed in three ways. Firstly, quantitatively, by calculating the width of the interval about $Cost_{asis}$ which results from regarding each AA_I as an interval. Secondly, qualitatively, by showing the number of functions in the project with each development status. Thirdly, also qualitatively, by breaking down the number of man-weeks of engineering effort assigned to the project according to development status.

The case study did not describe any formal prioritisation or risk control practices.

Conclusions

The risk of an under-estimate of elapsed time (possibly resulting in payment of a penalty clause for example) is not identified. Neither are the risks of integration costs or development costs having been under-estimated.

An opportunity which is not formally identified and analysed in the case study is the potential benefit of performing development work on functions. An estimate of the likely volume of sales for a function after some development work has been performed, taken in conjunction with the development costs and the difference in application costs before and after the development work, could be used to estimate the opportunity for that particular function. This cost opportunity could be aggregated over all functions in the project and weighed against the cost risk.

The use of probabilistic techniques (such as Monte Carlo simulation) rather than modelling each function cost as an interval, could be used to obtain an indication of the likelihood of different values of the aggregated costs, rather than the entire range of possible costs.

If the functions were prioritised, according to which contributed the most towards the overall project cost risk, the early development work could be targeted towards those functions which were making the most significant contribution to the project cost risk. A risk control strategy of continual monitoring of function status and reporting of overall project risk could also be of potential benefit both before and after the tender has been produced.

4.3.2 Perceptions of Risk at Rover

This assessment is based on a workshop involving a design team-leader, a purchasing team-leader and an expert from the vehicle cost estimation department. Partial transcripts are provided in Appendix C. Information was also obtained from informal discussions with Neil Davis of the Warwick Manufacturing Group at Warwick University, from various documents written by him and also from an internal Rover document entitled "Effective Cost Management Guidelines", part of which is reproduced later in this section. The design project under consideration is an automotive facia, although risk management for the entire vehicle is discussed.

For the purposes of this document, the early design phase is considered to consist of the *product development period* (see [Davis 1994]) up to *the D0 event* which is the point in time when programme approval is obtained and capital expenditure is sanctioned. By D0 the suppliers are contractually bound to an agreed cost target for each component. Prior to D0, the vehicle is decomposed into *areas*, each of which is decomposed into Vehicle Parts Groups (*VPG*s), each of which is in turn decomposed into components, known as Part Number Groups (*PNG*s). The decomposition is illustrated in Figure 4-12. This information is reproduced from [Davis 1994] which also gives further detail of the product definition mechanisms used at Rover, particularly the feature-based decomposition which is not considered relevant to this section, except to say that each PNG must support a set of features which are identified in the Product Development Sheet (*PDS*).

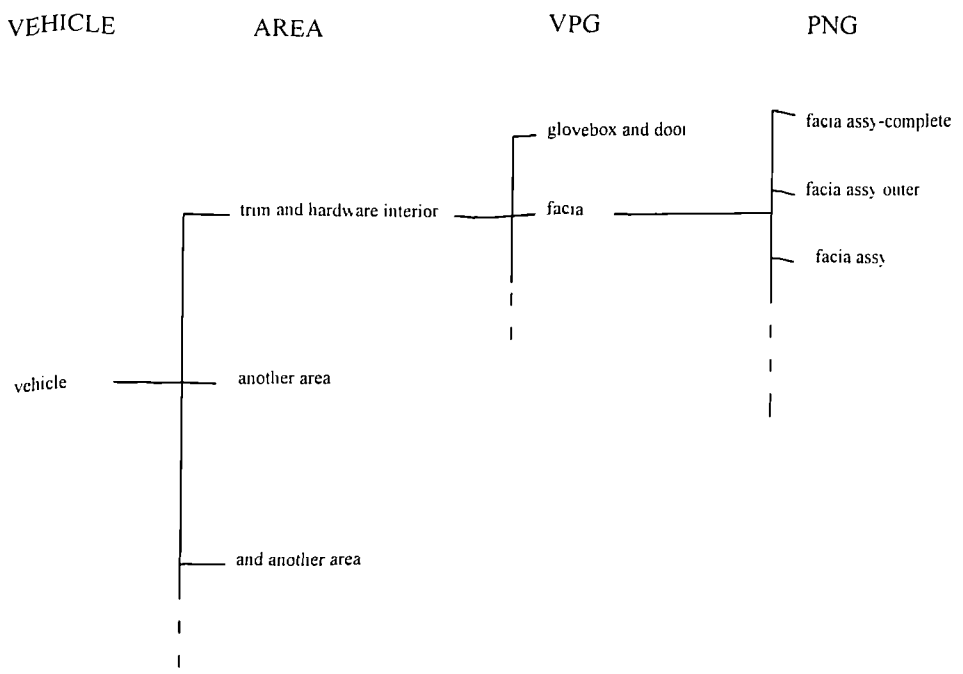


Figure 4-12: Example vehicle decomposition during product development

During early design, estimates are obtained for the piece-part and tooling costs for each PNG. These are aggregated to provide estimates for each VPG, each area and the entire vehicle. During early design, cost targets are assigned at the PNG, VPG and area level and at D0 the cost targets are agreed by the suppliers.

Risk Identification

There are several separate categories of risk which are identified. Firstly, during product development the risk of an under-estimate (and the opportunity of an over-estimate) of the costs for the component as it is currently envisaged is identified. Secondly, and separately, the risk that the component will need to change in some way which increases its costs (and conversely the opportunity for the component to change in a way which reduces its cost) is identified. The example given in the transcript in Appendix C[1] is that there may be an opportunity to avoid painting a component, thus reducing the piece-cost.

As with Cegelec, it is implicit in the quantitative analysis (see below) of under/over estimate of risk that the impact on the project is assumed to be equal to the resulting change in cost.

A third category of risk which is identified is "economics" - changes in currency exchange rates, raw material costs and so on[2]. A contingency is built in to the cost estimates at a programme level to cope with these risks. This is discussed in the transcript in Appendix C, but is not considered further here.

Finally, a fourth category of risk which is identified[3] is the risk of an under-estimate caused by some components having been omitted from the cost-roll-up. The fixings (e.g. screws, nuts and bolts) are an

[1]See marked paragraph number 2 in Appendix C

[2]See marked paragraph number 5 in Appendix C

[3]See marked paragraph number 1 in Appendix C

example of components which are generally at too detailed a level to warrant inclusion in the costing at this early stage, but which typically contribute 2% or 3% of the piece part costs.

Risk Analysis

The likelihood of an over or under-estimate of component cost is determined by the source of the estimate (i.e. who provided it). It is assumed that the costs lie within an interval, whose limits are given by Table 4-6 (these percentages are reproduced from the workshop transcript in Appendix C[4]).

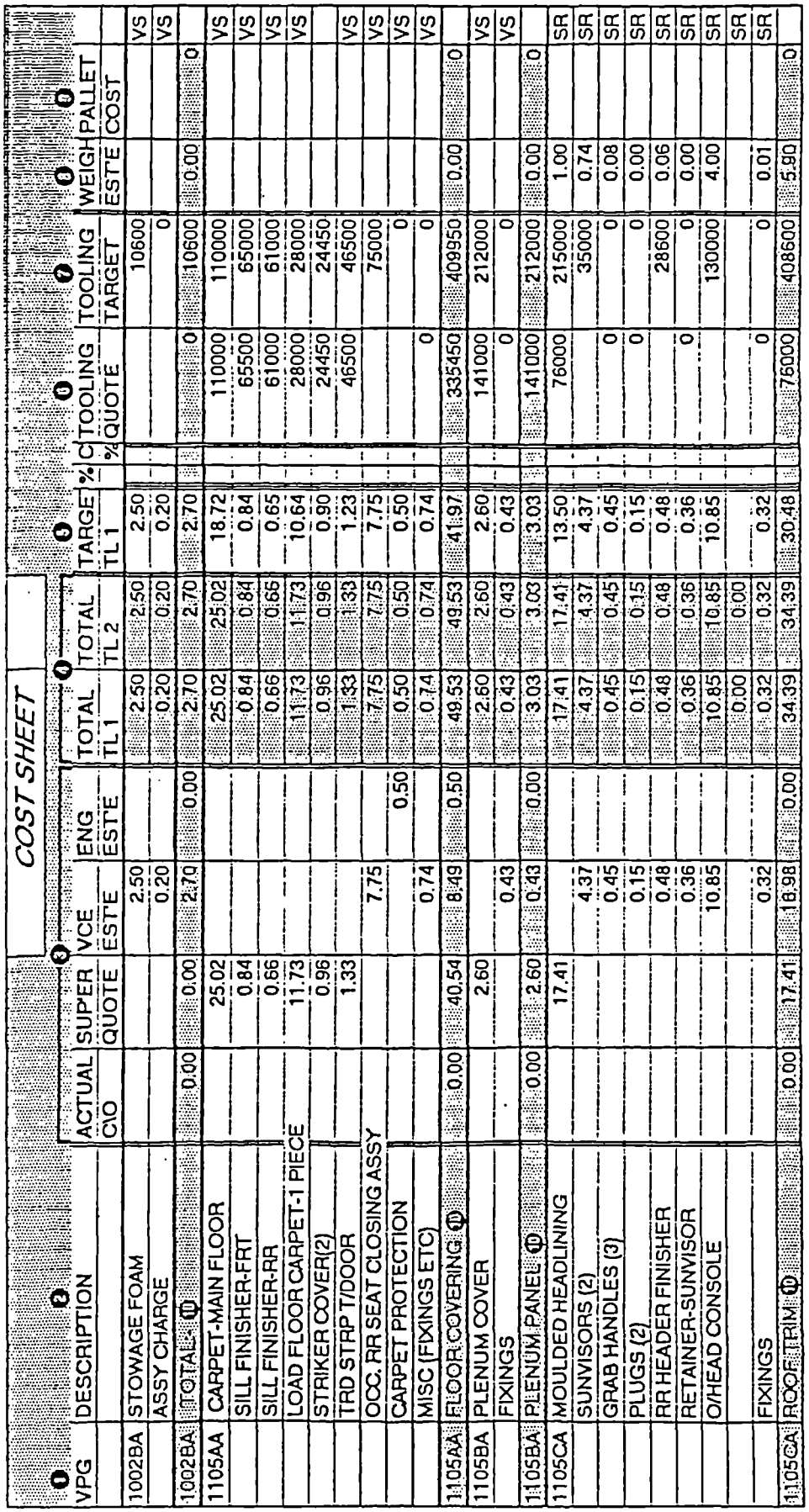
Source of estimate of PNG cost	Lower limit, as percentage of estimate	Upper limit, as percentage of estimate
Engineer's estimate	85%	115%
Vehicle cost estimation dept	90%	110%
Supplier quote	95%	105%
Previously manufactured component	100%	100%

Table 4-6: Percentage Limits According to Source of Estimate (Rover Group)

When a more certain estimate is obtained, it directly replaces the previous estimate. For example, the supplier quote will replace the Vehicle Cost Estimation (VCE) department estimate. These intervals are aggregated to give a maximum and minimum cost estimate for each area.

This is shown in the cost sheet in Figure 4-13 and cost summary sheet in Figure 4-14. In the cost sheet, each component (e.g. "stowage foam", shown at the top) has a single estimate recorded for its piece cost in one of four possible columns (marked as (3)) - an actual cost (for an existing component), a supplier quote, a VCE estimate or an engineer's estimate. The estimates in each column are aggregated for each VPG (shown shaded, for example "floor covering") - this tells us how much of the total piece cost estimate for a VPG originates from the VCE dept., how much from engineers' estimates etc. On the cost summary sheet, the totals for each VPG in an area are aggregated - once again, broken down according to the source of estimate. Percentage limits, representing the degree of uncertainty in the estimate, are then applied to the area piece part cost totals, yielding a cost interval for the whole area for each of the four estimate sources. These four intervals are added to yield the overall piece part cost interval for the area. During product development, the cost intervals for each area will gradually narrow, as increasingly accurate sources are obtained for the estimates.

[4]See marked paragraph number 3 in Appendix C

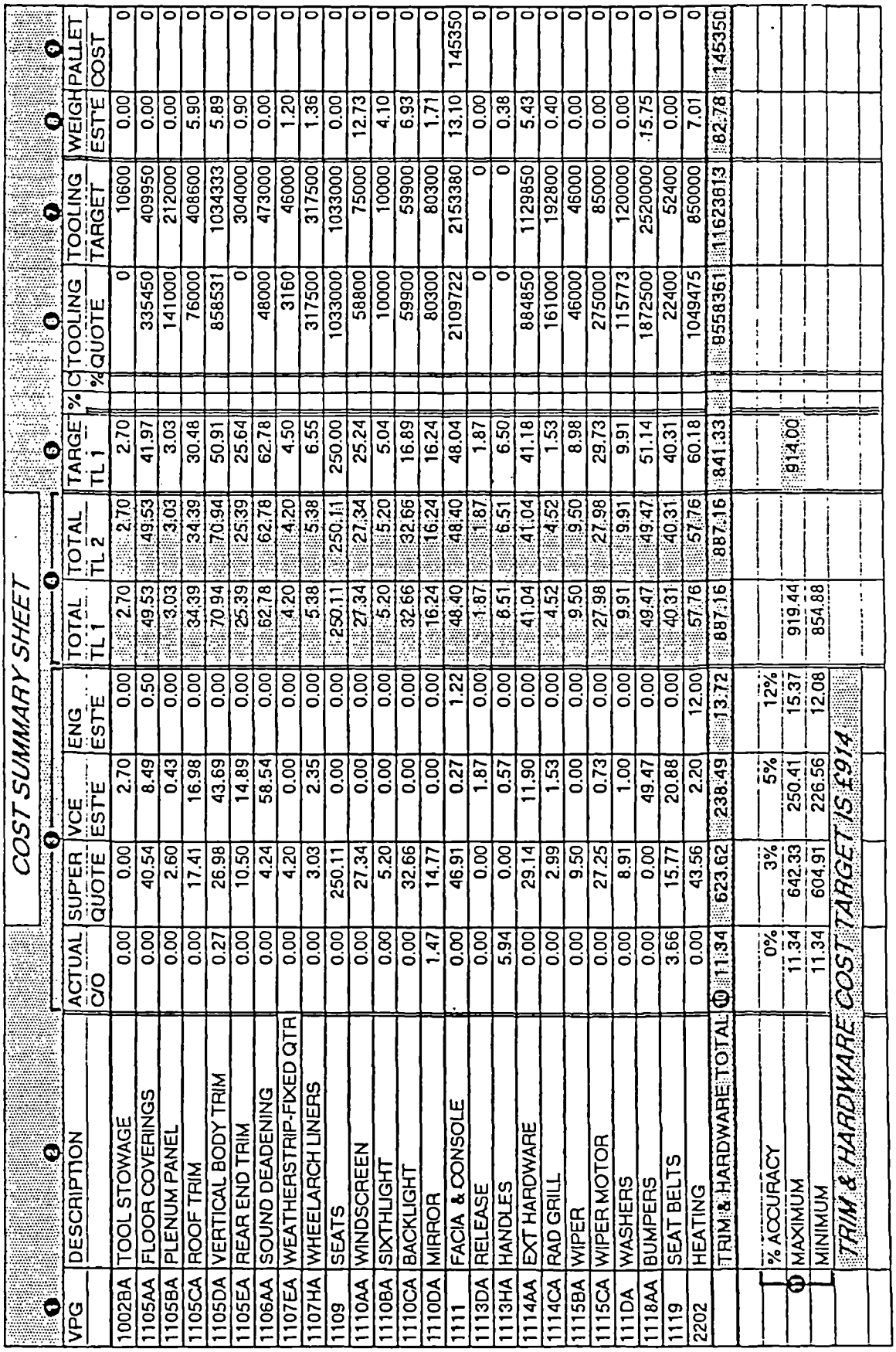


DATE 5/9/91-PAGE1

COST SHEET

VPG	DESCRIPTION	ACTUAL C\O	SUPER QUOTE	VCE ESTE	ENG ESTE	TOTAL TL 1	TOTAL TL 2	TARGE TL 1	% C	TOOLING QUOTE	TOOLING TARGET	WEIGH ESTE	PALLET COST	
1002BA	STOWAGE FOAM			2.50		2.50	2.50	2.50			10600			VS
	ASSY CHARGE			0.20		0.20	0.20	0.20			0			VS
1002BA	TOTAL- ①	0.00	0.00	2.70	0.00	2.70	2.70	2.70		0	10600	0.00	0	
1105AA	CARPET-MAIN FLOOR		25.02			25.02	25.02	18.72		110000	110000			VS
	SILL FINISHER-FRT		0.84			0.84	0.84	0.84		65500	65000			VS
	SILL FINISHER-RR		0.66			0.66	0.66	0.65		61000	61000			VS
	LOAD FLOOR CARPET-1 PIECE		11.73			11.73	11.73	10.64		28000	28000			VS
	STRIKER COVER(2)		0.96			0.96	0.96	0.90		24450	24450			
	TRD STRP T/DOOR		1.33			1.33	1.33	1.23		46500	46500			VS
	OCC. RR SEAT CLOSING ASSY			7.75		7.75	7.75	7.75			75000			VS
	CARPET PROTECTION				0.50	0.50	0.50	0.50			0			VS
	MISC (FIXINGS ETC)			0.74		0.74	0.74	0.74		0	0			VS
1105AA	FLOOR COVERING ①	0.00	40.54	8.49	0.50	49.53	49.53	41.97		335450	409950	0.00	0	
1105BA	PLENUM COVER		2.60			2.60	2.60	2.60		141000	212000			VS
	FIXINGS			0.43		0.43	0.43	0.43		0	0			VS
1105BA	PLENUM PANEL ①	0.00	2.60	0.43	0.00	3.03	3.03	3.03		141000	212000	0.00	0	
1105CA	MOULDED HEADLINING		17.41			17.41	17.41	13.50		76000	215000	1.00		SR
	SUNVISORS (2)			4.37		4.37	4.37	4.37			35000	0.74		SR
	GRAB HANDLES (3)			0.45		0.45	0.45	0.45		0	0	0.08		SR
	PLUGS (2)			0.15		0.15	0.15	0.15		0	0	0.00		SR
	RR HEADER FINISHER			0.48		0.48	0.48	0.48			28600	0.06		SR
	RETAINER-SUNVISOR			0.36		0.36	0.36	0.36		0	0	0.00		SR
	O/HEAD CONSOLE			10.85		10.85	10.85	10.85			130000	4.00		SR
						0.00	0.00							SR
	FIXINGS			0.32		0.32	0.32	0.32		0	0	0.01		SR
1105CA	ROOF TRIM ①	0.00	17.41	16.98	0.00	34.39	34.39	30.48		76000	408600	5.90	0	

Figure 4-13: Cost Sheet: extract from "Effective Cost Management Guidelines" (internal Rover document)



DATE 5/9/91-PAGE1

COST SUMMARY SHEET

VPG	DESCRIPTION	ACTUAL C/O	SUP'ER QUOTE	VCE EST'E	ENG EST'E	TOTAL TL 1	TOTAL TL 2	TARGE TL 1	% C	TOOLING % QUOTE	TOOLING TARGET	WEIGH EST'E	PALLET COST
1002BA	TOOL STOWAGE	0.00	0.00	2.70	0.00	2.70	2.70	2.70		0	10600	0.00	0
1105AA	FLOOR COVERINGS	0.00	40.54	8.49	0.50	49.53	49.53	41.97		335450	409950	0.00	0
1105BA	PLENUM PANEL	0.00	2.60	0.43	0.00	3.03	3.03	3.03		141000	212000	0.00	0
1105CA	ROOF TRIM	0.00	17.41	16.98	0.00	34.39	34.39	30.48		76000	408600	5.90	0
1105DA	VERTICAL BODY TRIM	0.27	26.98	43.69	0.00	70.94	70.94	50.91		858531	1034333	5.89	0
1105EA	REAR END TRIM	0.00	10.50	14.89	0.00	25.39	25.39	25.64		0	304000	0.90	0
1106AA	SOUND DEADENING	0.00	4.24	58.54	0.00	62.78	62.78	62.78		48000	473000	0.00	0
1107EA	WEATHERSTRIP-FIXED QTR	0.00	4.20	0.00	0.00	4.20	4.20	4.50		3160	46000	1.20	0
1107HA	WHEELARCH LINERS	0.00	3.03	2.35	0.00	5.38	5.38	6.55		317500	317500	1.36	0
1109	SEATS	0.00	250.11	0.00	0.00	250.11	250.11	250.00		1033000	1033000	0.00	0
1110AA	WINDSCREEN	0.00	27.34	0.00	0.00	27.34	27.34	25.24		58800	75000	12.73	0
1110BA	SIXTHLIGHT	0.00	5.20	0.00	0.00	5.20	5.20	5.04		10000	10000	4.10	0
1110CA	BACKLIGHT	0.00	32.66	0.00	0.00	32.66	32.66	16.89		59900	59900	6.93	0
1110DA	MIRROR	1.47	14.77	0.00	0.00	16.24	16.24	16.24		80300	80300	1.71	0
1111	FACIA & CONSOLE	0.00	46.91	0.27	1.22	48.40	48.40	48.04		2109722	2153380	13.10	145350
1113DA	RELEASE	0.00	0.00	1.87	0.00	1.87	1.87	1.87		0	0	0.00	0
1113HA	HANDLES	5.94	0.00	0.57	0.00	6.51	6.51	6.50		0	0	0.38	0
1114AA	EXT HARDWARE	0.00	29.14	11.90	0.00	41.04	41.04	41.18		884850	1129850	5.43	0
1114CA	RAD GRILL	0.00	2.99	1.53	0.00	4.52	4.52	1.53		161000	192800	0.40	0
1115BA	WIPER	0.00	9.50	0.00	0.00	9.50	9.50	8.98		46000	46000	0.00	0
1115CA	WIPER MOTOR	0.00	27.25	0.73	0.00	27.98	27.98	29.73		275000	85000	0.00	0
111DA	WASHERS	0.00	8.91	1.00	0.00	9.91	9.91	9.91		115773	120000	0.00	0
1118AA	BUMPERS	0.00	0.00	49.47	0.00	49.47	49.47	51.14		1872500	2520000	15.75	0
1119	SEAT BELTS	3.66	15.77	20.88	0.00	40.31	40.31	40.31		22400	52400	0.00	0
2202	HEATING	0.00	43.56	2.20	12.00	57.76	57.76	60.18		1049475	850000	7.01	0
	TRIM & HARDWARE TOTAL	11.34	623.62	238.49	13.72	887.16	887.16	841.33		8558361	11623613	82.78	145350
	% ACCURACY	0%	3%	5%	12%								
	MAXIMUM	11.34	642.33	250.41	15.37	919.44		914.00					
	MINIMUM	11.34	604.91	226.56	12.08	854.88							
	TRIM & HARDWARE COST TARGET IS £914												

Figure 4-14: Cost Summary Sheet: extract from "Effective Cost Management Guidelines" (internal Rover document)

The likelihood of a change in the component causing a change in cost, and the impact of this change on the project, are assessed in a more qualitative fashion. Risks and opportunities are recorded for each PNG in units of cost. For example, a risk of £1 and an opportunity of £2 on the piece-part cost of a PNG which is estimated as £10 implies that there's a risk it will cost £11 and a possibility it will only cost £8. There may be several risks or opportunities identified for each PNG. Thus each component cost should be modelled as a discrete random variable, not an interval, with the difference between one discrete value and the next representing the realisation of a single risk or opportunity. The risk and opportunity costs are not aggregated to the VPG or area levels. Instead, they are used by a panel of experts (including people from design, purchasing, VCE and suppliers) to judge a suitable adjustment to the target cost which is eventually (at D0) agreed with the supplier. This is an iterative and qualitative process. Thus the impact of these risks and opportunities on the project are not quantified, but are "insured against" in the cost targets agreed - a contingency is effectively built in at a PNG level and is not propagated up the hierarchy[5].

The risk of component omission has traditionally been dealt with by including a global "fixings" figure for each VPG, estimated from the nearest existing VPG. The risk that this is an under-estimate is not quantified.

Prioritisation

No quantitative prioritisation techniques have been identified - although the cost sheets and cost summary sheet shown in Figure 4-13 could be used to determine which areas and VPGs are contributing the most towards the uncertainty in the estimates of the area costs.

Risk Control, Resolution

The cost under/over estimate risk is resolved through the process of gradually obtaining increasingly accurate quotes for each PNG. No formal mechanism for resolving the risks and opportunities associated with possible changes in the component has been identified.

Risk Control, Monitoring and Reporting

The estimated costs are reported once a month as *Latest Indicated Cost* figures from the design team.

Conclusions

Although risks and opportunities are continually recorded at a component level during the design process, the impact of a component change on other components is not modelled. We assume that the relative complexity of this task is the reason that component risks and opportunities are not aggregated up the hierarchy. However, if these dependencies were modelled, the resultant risk analysis could provide valuable input to the expert panel when they meet to set the cost targets. Such a model would be able to model risks whose impact on the project was not simply the immediate cost increase/decrease.

Where uncertain information is recorded, it is stored as intervals rather than probability distributions. This is unsurprising considering the survey results presented in [Davis 1994B], which found that only 8% of respondents used numeric data in the form of probability distributions, with 21% using min-max or

[5] See marked paragraph number 4 in Appendix C

toleranced data. In the case of cost risk, however, since the most likely value is also known, the same data could usefully be interpreted as a triangular probability distribution.

It may prove beneficial to allow other factors than the source of a cost estimate to influence the degree of uncertainty associated with it. For example, an engineer could make an early cost estimate when his confidence was less than the +/- 15% assumed at present. Including such highly uncertain data in the model might permit costs to be rolled-up earlier in the project than at present. Also, it may sometimes be more appropriate to combine the engineer's and the VCE's estimates, rather than always replacing the engineer's estimate.

At D0, when the suppliers' cost targets are fixed, no quantified risk information is available - the risk that the suppliers will fail to meet their cost targets is not modelled. Yet Rover's close relationship with their suppliers means that in the event of a supplier failing to meet his cost target, Rover will share the resultant cost burden, by re-engineering the component for example if necessary[6]. Thus it would seem likely to be beneficial for Rover to be able to model the overall project risk due to suppliers failing to meet their cost targets.

4.3.3 Conclusions

Opportunities exist for both partners to improve their risk management by identifying several new areas of cost and time risk. Examples which may be common to both partners include:

- elapsed time risk
- component omission
- integration costs
- failure to achieve predicted sales/marketing volumes

In the survey results presented in [Davis 1994B], respondents were asked to identify the three most valuable and the three most uncertain types of information in their work. The information type with the strongest correlation between value and uncertainty was sales/marketing volumes, which 17 out of 50 respondents identified as valuable and a similar number identified as uncertain. The most valuable information type was piece-part cost, identified by 26 respondents as valuable and by 10 as uncertain.

Neither Rover nor Cegelec use probabilistic techniques for their cost-risk analysis - both rely on intervals when propagating uncertainty. This has the disadvantage that there is no indication of likelihood obtained when the costs are rolled-up. Neither partner uses formal or quantitative techniques to prioritise risks and thus direct their work towards early risk reduction.

Neither partner formally models the effect which one risk/opportunity may have on others within the project, except by aggregation up a hierarchy - the risks and opportunities are assumed to be independent at the component level.

---

[6] See marked paragraph number 6 in Appendix C

Decision support tools may be of benefit to both partners - for example to support the Rover team in setting cost targets and to support Cegelec in selecting the level of development investment for an existing feature.

Generally, neither partner performs continuous monitoring and reporting of risk during the design process. (There is an exception to this in that Rover do monitor the increasing accuracy of their cost estimates according to source). A major benefit of introducing formal risk monitoring and reporting would be the possibility of re-use of the reported risk data in subsequent projects to increase the accuracy of risk estimates.

4.4 Summary and Outline of Requirements for Risk Assessment Tool

4.4.1 Summary

In the first section of Chapter 3, theories and models of the design process were reviewed. It was concluded in particular that re-use of familiar solutions (as opposed to the generation and testing of all potential design solutions) is an integral part of the design process. This can be regarded as a strength rather than a weakness. The ideas of *set-based concurrent engineering*, where sets of alternative designs are concurrently and co-operatively developed, and of design as a process of *refinement* were also introduced. Refining processes include the addition of information, decreasing levels of abstraction and movement from the approximate to the precise.

In the second section of Chapter 3, a categorisation according to nature and subject was proposed for the types of uncertain information which are important during early design. The nature of such information was categorised as vague, numerically or categorically imprecise/uncertain, abstract, incomplete or inconsistent. The subject was categorised according to the information models in the transformation model of design proposed by McMahon *et al.* The types of uncertain information which should be representable in a risk model for early design were identified as:

- Numerical and categorical uncertainty, abstraction and incompleteness in the design environment model/s (such as uncertainty concerning usage conditions for the designed artefact).
- Numerical and categorical uncertainty, abstraction and incompleteness in the explicit design model/s (such as numerical uncertainty in dimensions or categorical uncertainty between alternative materials under consideration).
- Numerical uncertainty which is introduced through the use of heuristic and approximate methods in auxiliary model/s.
- Categorical uncertainty concerning alternative auxiliary models which may be used to evaluate implicit attributes (such as performance parameters, cost or weight).

Then, in the first two sections of Chapter 4, the risk in early design projects (i.e. arising from such uncertain information) was considered. The methodologies, tools and techniques for project risk management which are currently advocated or which are the subject of research were reviewed and the eight key elements of a risk management strategy were identified which the risk model must support:

1. goal and base plan definition
2. risk identification
3. risk impact and probability evaluation
4. risk prioritisation
5. modelling risk relationships
6. mitigation and contingency planning
7. allocating and continually monitoring budget levels
8. continual risk monitoring

The importance of taxonomies and checklists for risk identification was emphasised and some recent research work on knowledge-based approaches was described. The need for tools to facilitate rapid risk model building was identified. The third section described current practice in risk assessment and perceptions of risk at two of the collaborating industrial partners and concluded that current practice fell some way short of that prescribed by the methodologies described in the previous sections.

4.4.2 Outline of Requirements

Appendix A contains the requirements specification document which was produced for the risk assessment tool along with information about the extent to which these requirements were met. A high level overview of the most important requirements is presented in this section.

The aim of the risk assessment software tool was to allow the users (designers and others involved in the design process) to build a *risk model* which captured all of their uncertain knowledge about a design, during the early stages of the design process. The risk tool would then be used to evaluate this model and produce an assessment of the overall risks - these could be performance parameter risks, cost risk or time risk. It was considered necessary that the risk model could evolve to reflect the development of the design itself, in order that risk could be continually monitored. Thus it was an important requirement that the process of building the risk model should be made as fast and as simple as possible, so as to minimise the time taken from actually designing. Re-use of existing knowledge concerning risk in the particular design domain of the user company would not only expedite rapid model building but would also help to support the identification of as many risks as possible, in a similar way to the knowledge based approaches described earlier. Knowledge which would be re-used about a particular type of designed product would include previously identified lists of risks (checklists), approximate (heuristic) rules for estimating cost, performance parameters and other attributes and their risks and also historical information about such attributes.

The obvious way to encapsulate this sort of knowledge was as an object class (see Chapter 6 for a description of the principles of object-orientation). Therefore it was required that the risk model would be object-oriented. The risk model would consist of a database of instances of classes (i.e. objects) with relationships between them. The evolution of the model during the process of design would be a process of refinement involving the creation of new objects as increasing detail was added to the model and the modification of object attributes and deletion of objects as design decisions were taken and as approximate information was replaced by more precise information. A pre-defined set of object classes, termed the *base-classes*, would be

supplied with the tool, but user companies would be able to add their own classes which encapsulated knowledge about their particular design domain. It was anticipated that teams of co-operating designers would build and modify the design model. A modeller would be responsible for creating new specialised classes and for modifying the existing class definitions. It was anticipated that, initially at least, a single modeller would be responsible for building all the classes. Designers would then insert instances into the design model.

It was required that the tool could be used at any time to produce estimates of the overall risk in the project from the uncertain information stored in the risk model. The tool was also required to identify which sources of uncertainty in the risk model had the largest effect on the overall risk (i.e. to perform a *risk sensitivity analysis*) thus providing support for risk prioritisation.

The objects stored in the risk model would represent both the designed artefact and the design environment. Numerical uncertainty would be represented by permitting attributes to take as their values either probability distribution functions with their parameters or numerically defined probability distributions. Categorical uncertainty in the artefact model would be represented by including in the risk model sets of alternative designs under consideration; only one of which would eventually be chosen - as advocated by the proponents of set-based concurrent engineering. The same mechanism would be used to represent categorical uncertainty in the design environment. Incompleteness would be represented by permitting missing attribute values and missing objects. Abstraction would be represented by the use of general classes; for example, an object of the general class PowerPack might be replaced later by an object of the more specific class NickelCadmiumBattery, thus reducing the level of abstraction.

When building the classes, the modeller would define methods for deriving attribute values. It was required that these methods would include parametric expressions, rule-based methods and also approximate, heuristic methods. The numerical uncertainty in the auxiliary models would be represented by the uncertainty introduced by these heuristic methods. It was required that several different methods could be defined for deriving a single attribute value, providing alternative ways to evaluate attributes as more information became available. The most accurate method for which the input data was available would automatically be selected by the tool. Thus categorical uncertainty concerning alternative auxiliary models which may be used to evaluate implicit attributes would be incorporated into the risk model. It was also required that the class definition would include default values for attributes - thus a default value of UNKNOWN (or missing) would effectively serve as an entry on a risk identification checklist. It was required that constraints on attributes, such as budgetary constraints, would be represented in the risk model and that the tool would be able to evaluate the probability that a constraint, or set of constraints, would be met.

An important requirement, which is not recorded in Appendix A because it did not become apparent until the prototype tool was tested, was that the tool should support modelling of variants and "configuration design". Variants are alternative components and assemblies, only one of which is included in a particular product offering, but all of which will be manufactured - the customer will choose between the offered products.

To summarise, it was required that the risk tool and model should be able to represent the types of uncertain design information identified in Chapter 3 and that it should be capable of supporting (but should not prescribe) the types of risk management process described in Chapter 4.

4.5 References

[Albrecht 1979] Albrecht, A. J., "Measuring Application Development Productivity", *Proceedings IBM Applications Development Symposium*, Monterey, CA, pp. 83-92, October 1979.

[Boehm 1991] Boehm, B. W., "*Software Risk Management: Principles and Practices*", IEEE Software, (January 1990)

[Boehm 1981] Boehm, B. W., "*Software Engineering Economics*", Englewood Cliffs, NJ : Prentice Hall, 1981.

[Carr 1993] Carr, M. J., Konda, S. L., Monarch, I., Ulrich, F. C. and Walker, C. F., "Taxonomy-Based Risk Identification", *Technical Report CMU/SEI-93-TR-6*, Software Engineering Institute, Carnegie Mellon University, Pennsylvania, June 1993.

[Carter 1994] Carter, B., Hancock, T., Morin, J-M., and Robins, M., "*Introducing RISKMAN Methodology: The European Project Risk Management Methodology*", Oxford, UK: NCC Blackwell Ltd, 1994.

[Chapman 1979] Chapman, C. B., 1979, "*Large Engineering Project Risk Analysis*", IEEE Transactions on Engineering Management, vol. EM-26, no. 3, pp. 78-87, 1979.

[Chapman 1997] Chapman, C. B. and Ward, S., 1997, "*Project Risk Management: Processes, Techniques and Insights*", Chichester : John Wiley & Sons Ltd, ISBN 0-471-95804-2, 1997.

[Charette 1989] Charette, R., "*Software Engineering Risk Analysis and Management*", New York : McGraw-Hill, 1989.

[Cooper 1987] Cooper, D. F. and Chapman, C., "*Risk Analysis for Large Projects: Models, Methods and Cases*", Chichester : John Wiley and Sons, ISBN 0 471 91247 6, 1987.

[Cowderoy 1995] Cowderoy, A., "GOAL Risk Tool, Overview", *GOAL/CITY-DOC-W5-185-R1*, City University, London, UK, January 1995.

[Davis 1994] Davis, N., "Model:ProdDefinition", *project study report STARTED ROV/003/1*, June 1994.

[Davis 1994B] Davis, N., "Group Cost Management Survey: Questionnaire Analysis", Issue 1, *internal report* Warwick University, UK, August 1994.

[Eeckhoudt 1995] Eeckhoudt, L. and Gollier, C., "*Risk evaluation, management and sharing*", New York : Harvester Wheatsheaf, ISBN 0-7450-1592-1, 1995.

[Keeney 1976] Keeney, R. L., and Raiffa, H., "*Decisions with multiple objectives: preferences and value tradeoffs*", New York: John Wiley & Sons, 1976.

[Klein 1993] Klein, J. H., "Modelling Risk Trade-Off", *Journal of the Operational Research Society*, vol. 44, no. 5, pp. 445-460, 1993.

[Klein 1994] Klein, J. H., Powell, P. L., and Chapman, C. B., "Project Risk Analysis Based On Prototype Activities", *Journal of the Operational Research Society*, vol. 45, no. 7, pp. 749-757, 1994.

[May 1993] May, J., Hall, P., Zhu, H., Cockram, T., Bird, N., and Winsborrow, L., "Fault Prediction for Software Development Processes", *proceedings of IMA conference on Maths of Dependable Systems*, Egham, Surrey 1993, 1993.

[Moder 1970] Moder, J. J., and Phillips, C. R., 1970, "*Project Management with CPM and PERT*", New York: Van Nostrand, 1970.

[Monarch 1995] Monarch, I. and Gluch, D. P., "An Experiment in Software Development Risk Information", *Technical Report CMU/SEI-95-TR-014*, Software Engineering Institute, Carnegie Mellon University, Pennsylvania, October 1995.

[Niwa 1981] Niwa, K., "Trouble Library", *The Japan Society of Industrial Machinery Manufacturers, A Joint Study for Foreign Project Risk Management*, p. 114, 1981.

[Niwa 1989] Niwa, K., "*Knowledge-Based Risk Management in Engineering: A Case Study in Human-Computer Cooperative Systems*", New York : John Wiley & Sons, ISBN 0-471-62893-X, 1989.

[Ould 1990] Ould, M. A., "*Strategies for Software Engineering: The Management of Risk and Quality*", Wiley : Chichester, 1990.

[Pressman 1992] Pressman, R. S, "*Software Engineering, A Practitioner's Approach*", Third Edition, 1992.

[Putcha 1991] Putcha, C. S. and Rao, S. J. K., "Probabilistic network analysis for project completion time", *Civil Engineering Systems*, vol. 8, pp. 179-184, 1991.

[Rao 1993] Rao, S. J. K.; Kreiner, J. H. and Putcha, C. S., "Total Quality Engineering, Failure Analysis and Stochastic Process Evaluation of Engineering Project Alternatives". *IEEE*, pp. 275-280, 1993.

[Rosenblueth 1981] Rosenbleuth, E. "Two-Point Estimates in Probabilities", *Applied Mathematical Modelling*, vol. 5, October, 1981.

[Sisti 1994] Sisti, F. J. and Joseph, S, "Software Risk Evaluation Method: Version 1.0", *Technical Report CMU/SEI-94-TR-19*, Software Engineering Institute, Carnegie Mellon University, Pennsylvania, December 1994.

[Smith 1995] Smith, R. P., "Managing Risk by Reordering Tasks in Engineering Design", *Proceedings Design Engineering Technical Conferences*, ASME 1995, Boston, USA, vol. 2, pp. 585-591, 1995.

[Watson 1994] Watson, R. M., "Case Study of Cost/Risk factors in Tendering", *project study report STARTED/CEG/965/2*, June 1994.

[Williams 1995] Williams, T., "A Classified Bibliography of Recent Research Relating to Project Risk Management", *European Journal of Operational Research*, vol. 85, pp. 18-38, 1995.

CHAPTER 5

The Representation of Uncertainty

In this chapter, the choice of suitable representations for uncertainty in the model is described. In the first section a brief review is presented of the representations for uncertain and incomplete information which are currently used in engineering, including sets of point values, interval-based, probabilistic and fuzzy models and mass assignments and pairs of measures (support pairs, belief intervals and probability intervals). The techniques used to propagate uncertainty in such models and their application areas in engineering design are described in more detail in Appendix B, along with techniques for finding the "closest match" and for identifying the "best" amongst a set of alternatives. The second section briefly compares probabilistic representations with fuzzy models in particular, and explains why a probabilistic representation was chosen for the risk model.

5.1 Representations for Uncertain and Incomplete Information

In this section the representations available for uncertain and incomplete information are briefly reviewed, with examples of their application in engineering design. The section begins with the simplest case where the only information available concerning a variable is the set of possible values it may take; such information is represented by a discrete set or an interval. The representations offered by probability theory and by fuzzy sets are then developed. Finally, the relationship of mass assignment theory (a generalisation of probability theory) to support pairs, belief intervals and probability interval representations is briefly described.

5.1.1 Discrete Sets and Intervals

Suppose a choice between a set of discrete options has not yet been made. This is an example where the simple existence of diversity, in itself, indicates the presence of uncertainty. The simplest example of a data representation for uncertain information is thus a set of discrete values. The set of orthogonal experiments performed in Taguchi class experimental design (see Appendix B) could be regarded as an example of uncertainty due to diversity - the basic Taguchi method effectively assumes all values of control and noise factors are equally likely, although Otto and Antonsson have recently suggested a method for incorporating probability into the Taguchi method ([Otto 1993] and Appendix B). Another example of uncertainty simply due to diversity is provided by multivariate data-sets from which nearest-neighbour, or closest-match, is required where the data itself is represented by point values.

The simplest method for modelling uncertainty in a continuous parameter value is to represent the parameter by its maximum and minimum possible values, with no information concerning the likelihood or probability or preferability of any particular value within the range. If no such information is available, an interval will

be the most appropriate representation for the parameter. A variation on the simple interval representatic the nominal value plus limits, used in specifying engineering dimensions (for example, $50\ ^{+0.10}_{-0.05}$). Su representation can be considered as an interval (in the example, [49.95 50.10]), augmented by specifica of the nominal, or target, value (in the example, 50).

Interval analysis was originally developed to analyse the effect of instrument resolution and inaccurac measurements. In the 1960s, with increasing use of computers, interval analysis found a new applicatio calculating the effect of machine resolution on computed quantities (see [Moore 1966] and Appendix besides interval arithmetic, Moore also presents techniques for calculating derivatives and integral interval functions and thus also Taylor expansions). The main application of simple interval analysi Engineering Design until recently has been in worst-case tolerance analysis. More recently, Ward Seering have developed a calculus of "labelled intervals" which is used for making quantitative inference their "design compiler", a computer tool for design automation which constructs an optimal design fro schematic and a specification supplied by the user, and a catalogue of available components. The set components in the catalogue and the user's specification are described using a labelled interval specifica language. These applications are described in the Section entitled "Interval Analysis" in Appendix B.

5.1.2 Probability Theory

The term "probability" is generally accepted as meaning "likeliness to be true". In [Ayer 1973], the au gives three categories of statements about probability - statements of *a priori* probability which relate to calculus of chances (for example "the probability of a fair coin landing head side up is 1/2"), statis statements which relate to the frequency of occurrence of an event (for example "the probability that it rain in Bristol tomorrow during the morning rush-hour is 1/12") and statements concerning degree of b (for example "it is highly probable that there will be a permanently manned station on the moon before end of the 21st century").

In his "Treatise on Probability" [Keynes 1921], Keynes describes how the theories which characte probability in each of these three ways arose historically. It is interesting to reproduce part of his argun here, since the division between these three views of probability persist to the present day.

Keynes states that the earliest attributable theory of probability is Aristotle's - that what is probable is v has usually happened in the past. This is the basis of the frequentist or classical view of probability, arg by John Venn in his "Logic of Chance" in 1866, - that the probability of an event is a measure of frequently it has occurred in the past. This view was generally accepted by the 18th century philosop with the notable exception of Hume who argued that there is no valid basis for believing that what happened in the past will happen in the future and in his "Treatise of Human Nature" he opines that ' probable reasoning is nothing but a species of sensation".

Keynes argues that the theory of probability which was generally accepted by the mathematicians of time, however, was quite different. This was based upon the "Principle of Indifference" which states th we have no experience then we judge alternatives to be equally probable. Thus the probability of a partic

outcome may be assessed by counting how many equi-probable alternatives correspond to each outcome of interest. This is the basis of the calculus of chances, the now familiar arithmetic for predicting probabilities of combinations in games of chance, developed by Pascal and Fermat in the 17th century.

Bernoulli also based much of his work upon the Principle of Indifference and the idea of "counting chances". Bernoulli's "law of large numbers" states that for large numbers of trials, the frequency of a particular outcome will tend towards its probability. This provides a bridge between the "frequentist" and "calculus of chances" views of probability and allows the calculus of chances to be applied beyond games of chance, provided that there is a large amount of experience available.

Keynes held that the frequentist view is "a very grave departure from the established use of words; for it generally excludes a great number of judgements which are generally believed to deal with probability" (see [Keynes 1921, p. 95]). He is also quite scathing about the indiscriminate application of the Law of Indifference by mathematicians and philosophers, particularly in Laplace's "Law of Succession", up until the beginning of the 19th century. Keynes himself viewed probability as a logical relation (a relation between the evidence and the proposition) and he attempted to define this relation in his work "A Treatise on Probability". His definition of the process of probabilistic inference only permits a quantitative judgement of probability (as opposed to a qualitative ranking of alternatives) in the case where the Principle of Indifference holds - equiprobable, exclusive and exhaustive alternatives.

As Keynes pointed out, taken in its strictest sense, the frequentist view implies that it is only reasonable to discuss the probability of events about which we have experience. Also, in [Ayer 1973] the author points out that statistical information always relates to classes, rather than individuals, and the strict frequentist view of probability does not yield a result in the case that an individual belongs to two classes with different historical behaviour. Ayer illustrates this problem using the example of Petersen the Swede:

> *Petersen is a Swede and Swedes are probable to be Protestant. Petersen has been on pilgrimage to Lourdes and those who have made such a journey are not probable to be Protestant. That is as far as the strict frequentist approach takes us. In the frequentist view, there is nothing "better'" about basing the judgement of probability on the statistical information about the class of Swedes who have been to Lourdes - no one class offers a better estimate of probability than any other.*

The calculus of chances on the other hand does not allow us to make any statements about the probabilities of real events - it concerns idealised coins and perfectly shuffled packs of cards - it simply relates probabilities of complex outcomes (winning a card game) to probabilities of simple ones (drawing a particular card from an idealised well-shuffled pack).

It was Bernoulli who first proposed probability as a measure of belief or "degree of certainty" and Liebnitz who first argued that probability "is in proportion to what we know" [Skidelsky 1992, p. 77]. When such belief is considered to be relative to the body of evidence available to the holder of the belief, Bayes' famous theorem:

$$Pr(A|B) = \frac{Pr(A).Pr(B|A)}{(Pr(A).Pr(B|A) + Pr(NOT\ A).Pr(B|NOT\ A))}$$

can be used to modify the degree of certainty (that A is TRUE) in view of new evidence (about whether B is TRUE). This view, known as the Bayesian or personalist view, allows us to consider the probability of a much wider range of events than the frequentist view.

The foregoing describes the personalist and frequentist views of probability. In [Wood 1989], the authors present an alternative view of probability as a measure of the preference of the designer. Uncertain design parameters are represented as triangular probability distributions where the range of values with non-zero probability represents the interval of possible values and the value with the peak probability represents the value preferred by the designer. The classical probabilistic calculus is still used however to propagate the uncertainty.

In probability theory, an uncertain value is represented by a random variable. The random variable may be described by a single event probability value (for a logical random variable), a discrete set of probabilities (for a discrete variable) or a probability density function (PDF) (for a continuous random variable). Many probabilistic models require that uncertainty should be propagated through functions of continuous random variables. The commonly used techniques broadly fall into three categories: analytic methods (moment-based methods, special cases based on assumptions of independence or convenient PDFs which includes most reliability methods, weak probability statements based on Chebychev's Theorem [Freund 1980, p. 142]), simulation (Monte Carlo) and numerical methods (numerical convolution, discrete probability distributions, histogram representations), all outlined in Appendix B.

5.1.3 Fuzzy Sets

L.A. Zadeh developed the theory of fuzzy sets in the 1960s (see [Zadeh 1965]). In traditional (or "crisp") set theory, an element either belongs to a set or it does not. In fuzzy set theory, an element may have partial membership of a set. This allows the representation of vaguely defined sets such as "the set of tall people". Some people (say those over 185 cm tall) are definitely members of the set of tall people. Other people (say those less than 150 cm tall) are definitely not members of such a set. Most people will be partial members of the set, for example those who are "quite tall".

A fuzzy subset A of a universe of discourse U is defined by a membership function:

$$\mu_A : U \rightarrow [0,1]$$

which associates with each element of U a membership value between 0 and 1 where 0 represents non-membership and 1 represents definite membership. Intermediate values denote partial membership.

From the concept of a fuzzy set arose the concepts of the possibility distribution, fuzzy numbers (see [Evans 1989]) and linguistic variables. Possibility theory, developed by Zadeh in 1977, is a branch of fuzzy set

theory which provides the concept of the possibility distribution. The possibility distribution is a constraint upon the values which a variable may take. An interval for example is a "crisp" constraint, where the variable either may or may not take a particular value. A possibility distribution on the other hand is a "fuzzy" constraint, representing the idea that one value may be "more possible" than another.

To illustrate the difference between a possibility distribution and a probability distribution, consider the variable X which represents the number of replacement ball-bearings which a designer will choose to supply with a system he is designing. Suppose that cost is the only constraint he is operating under. Some values from the probability and possibility distributions for the variable X are given below:

x	0	1	2	3	4	5
Prob = Pr[X = x]	0	0.3	0.6	0.07	.03	0
Poss = $\mu_X(x)$	1	1	1	.7	.3	0

It can be seen that although it is perfectly possible for the designer to include no spare parts at all in the kit (since that will incur no extra piece-cost whatsoever), it is not probable. On the other hand, it is quite impossible for him to include 5 spare ball-bearings because that would definitely exceed the cost budget, so the probability that he will do so is also zero.

Thus, the possibility distribution for the variable X provides a measure of the *possibility* that X will take the value x for each x in the domain. Contrast this with the probability distribution which for each x provides a measure of the *probability* that X will take the value x.

A formal definition of the possibility distribution is given in [Zadeh 1978]. Let F be a fuzzy set over a universe of discourse U with membership function μ_F. Let X be a variable over U. The proposition "X is F", associating the variable X with the concept represented by F, induces a possibility distribution Π_X on X which restricts the values which X may take, where Π_X is defined by $\Pi_X(x) = \mu_F(x)$ for all $x \in U$.

In [Dubois 1980], a fuzzy number is defined to be a convex fuzzy set over the real numbers which takes a value of 1 somewhere. A fuzzy number could be used to represent a value of "about 3" for example. Fuzzy numbers may be propagated through functions using the extension principle (Appendix B). Wood and Antonsson have modelled design preference using fuzzy numbers and the extension principle (Appendix B), Allen, Mistree *et al.* have used fuzzy numbers to model uncertain constraints and goals in design optimisation (Chapter 3), Blockley has used fuzzy numbers in fatigue analysis (Appendix B) and Tee and Bowman and others have also applied the method of fuzzy weighted averages in civil engineering (Appendix B). Other work on fuzzy design optimisation is presented in [Díaz 1989] and [Otto 1991] .

A linguistic variable (see [Zadeh 1975]) takes values which are words or phrases. For example the linguistic variable *height* might take values such as *very_tall, tall, average, short* and so on. The linguistic variable has a base variable, for example *height_in_cm,* and each value, such as *tall*, restricts the base variable according

to a compatibility function, f_{tall}. The compatibility function provides a mapping from the base variable domain to the interval [0,1], and is similar to a membership function. For example, the compatibility function for the value *tall* might be defined by the straight line segments joining the points:

$$f_{tall}(190\ cm) = 1$$
$$f_{tall}(170\ cm) = 0.5$$
$$f_{tall}(140\ cm) = 0$$

where

$$f_{tall}(x\ cm) = 0 \quad \text{for all } x < 140\ cm$$
$$f_{tall}(x\ cm) = 1 \quad \text{for all } x > 190\ cm$$

Linguistic variables represent uncertainty in the definition of a variable's value, as opposed to imprecision. Uncertainty may be propagated through linguistic variables using semantic unification, as explained in the section entitled "Support Logic" in Appendix B. Linguistic variables have been applied in fuzzy database storage and retrieval (Appendix B).

To summarise, fuzzy set based concepts extend the types of uncertainty which we may represent beyond crisp possibility (intervals) and likelihood (probability). Linguistic variables may be used to represent uncertainty in the definition of a variable's value, fuzzy numbers and possibility distributions may be used to represent elastic constraints on the values which a variable may take.

5.1.4 Mass Assignments and Pairs of Measures

In the sections above, representations are described for the cases where the distribution (possibility or probability) is either completely unknown (in which case an interval representation is chosen) or is known over the entire universe of discourse. However, the probabilities may only be known for some subsets of the universe of discourse. In this case a distribution known as a *mass assignment function* may be defined. In addition to extending the types of uncertainty which we may represent, measure theory (based on mass assignment functions) provides a general framework within which both probability and possibility may be viewed.

In the area of uncertain knowledge representation in artificial intelligence, the set which the distribution is defined over (corresponding to the universe of discourse for a fuzzy set membership function) is known as the frame of discernment and is finite. In Dempster and Shafer's formulation of the theory, the frame of discernment, U, consists of a set of mutually exhaustive and exclusive hypotheses.

A mass assignment function is a function m which maps P(U) (the set of all subsets of U, known as the power set of U) onto the interval [0,1]:

$$m: P(U) \rightarrow [0,1]$$

such that:

$$\sum_{A \in P(U)} m(A) = 1$$

and:

$$m(\varnothing) = 0$$

Then m(A) can be interpreted as the amount of belief committed exactly to A. Note that this is not the same as the amount committed to A and all of its subsets. If we wish to represent the amount of belief that a particular element of interest (for example the one "true" hypothesis) lies in the set A, then we have to use Dempster-Shafer's belief function:

$$Bel(A) = \sum_{B \subseteq A} m(B)$$

since if the element of interest lies in B then it also lies in all supersets of B.

In mass assignment theory, a function g: $P(U) \rightarrow [0,1]$ which satisfies the two axioms

A1: (boundary condition) $g(\varnothing) = 0$ and $g(U) = 1$

A2: $g(A_1 \cup A_2 \cup ... \cup A_n) \geq \Sigma_i\, g(A_i) - \Sigma_{i<j}\, g(A_i \cap A_j)$
$+ ... + (-1)^{n+1} g(A_1 \cap A_2 \cap ... \cap A_n)$

for every $n \in \mathbf{N}$ (where **N** signifies the natural numbers) and for every collection of subsets of U.

is known as a *necessary support measure*. Thus for two subsets A and B of U, axiom A2 states that:

$$g(A \cup B) \geq g(A) + g(B) - g(A \cap B)$$

It can be shown that every necessary support measure may be expressed, in terms of a mass assignment function, as a belief function Bel as defined above.

For a *probability measure* (also known as a *Bayesian belief measure*), axiom A2 is replaced by the stronger axiom A2':

A2': (additivity) $g(A \cup B) = g(A) + g(B)$ whenever $A \cap B = \varnothing$

(Axion A2' is one of the three Kolmogorov axioms of probability). Thus a probability measure is a special case of a belief function. Note that g itself is not a probability distribution - it is defined over P(U), not U. However, it can be shown that a belief function Bel is a probability measure on a finite power set if and only if its mass assignment function maps all non-singletons to zero. Thus a probability measure can be represented by a probability distribution.

A *fuzzy measure* on the other hand, does not have to satisfy axiom A2 and both necessary support measures and probability measures are special cases of fuzzy measures. Given a necessary support measure Sn, a *possible support measure* Sp, may be defined by:

$$Sp(A) = 1 - Sn(\overline{A})$$

Sn(A) represents the necessary (i.e. minimum) support for the element of interest lying in A. Thus Sn($\overline{A}$) represents the necessary support against the element of interest lying in A and (1 - Sn($\overline{A}$)) represents the maximum support (i.e. the possible support) for the element of interest lying in A. A support pair for A is then given by [Sn(A), Sp(A)]. In the Dempster-Shafer formulation, this is known as a belief interval.

The probability that the element of interest lies in A, Pr(A), lies within this interval. The width of the interval reflects the degree of uncertainty. Thus a belief interval or support pair may be regarded as a probability interval where:

[0,1] represents total ignorance i.e. total uncertainty
[0,0] represents certainty that the element of interest doesn't lie in A
[1,1] represents certainty that the element of interest does lie in A

In [Cui 1990], Cui and Blockley proposed an alternative calculus for evidential support pairs, termed **Interval Probability Theory**, which is not based on mass assignment theory but on the axioms of probability theory. The section entitled "Propagation techniques" in Appendix B describes two techniques for combining pieces of evidence expressed as belief function values:- **Support Logic** and **Dempster-Shafer evidence theory** and also describes Interval Probability Theory. Stone and Blockley have used Support Logic to build a knowledge-based system to re-use historical information for risk identification in civil engineering projects (Appendix B).

5.2 The Choice of Representations

In Chapter 4, the requirements for the risk model and tool were presented. The nature of the uncertain information to be represented was summarised as incompleteness, abstraction and numerical and categorical uncertainty. In this section the choice of a probabilistic representation for numerical uncertainty is justified, and then the ways in which categorical uncertainties are represented are explained.

5.2.1 Representing Numerical Uncertainty

In Chapter 2, risk was defined as a combination of likelihood and impact, and thus interval representations which contain no information concerning likelihood are inadequate for a risk model. The requirement to represent numerical information precludes logic-based and linguistic representations. The choice was therefore between fuzzy numbers (possibility distributions) or probability distributions. Comparisons between these two approaches have been published both in the engineering design literature and in the area of economic analysis. Proponents of the fuzzy approach often argue that probabilistic models are not appropriate to represent degree of belief, refusing to accept the personalist view of probability theory. In [Choobineh 1992], it is argued that probabilistic models are more widely used than interval, fuzzy or evidential models for historical reasons rather than because of their inherent appropriateness. The authors take a strictly frequentist view of probability:

> *An expert may feel that a given parameter is within a certain range and may even have an intuitive "feel" for the "best" value within that range. However, seldom will the expert have an empirical or even experiential foundation for the estimate based on frequency of occurrence. Indeed frequently the circumstances surrounding a given modelling situation are unique and irreplicable in practice. It is precisely the idea of frequency, however, that underlies the theoretical foundations of probability theory.* [Choobineh 1992, pp 907-908]

In [Thurston 1992] a probabilistic and a fuzzy approach to design evaluation are compared; utility theory is contrasted with fuzzy rating. As in previous work, the problem addressed is evaluation of design alternatives where there are several incommensurate, related and uncertain implicit design attributes for each alternative under consideration. If the relationship between the attributes is such that an improvement in one results in a degradation of another, such an evaluation of necessity involves a "tradeoff" between the different attributes. The application explored is the choice of material for use in an automotive bumper. The probabilistic approach taken is multi-criterion utility theory and this is contrasted with the use of a fuzzy weighted average. Each attribute of each alternative was assigned a value (or rating) which was a fuzzy number defined over the sub-set of the real number domain (0, 1). Membership functions were pre-defined for fuzzy numbers "very low", "low", "high" etc. Each attribute was also assigned a fuzzy weighting which was an indication of its importance to the designer. For each alternative, the extension principle was then used to calculate the membership function of the weighted average (the fuzzy rating approach is developed further in [Carnahan 1994]). Thurston and Carnahan concluded that fuzzy methods were preferable in preliminary design:

> *When the lack of precision or uncertainty has to do strictly with meaning or preference rather than frequency of occurrence, the mathematics of fuzzy sets is more appropriate than the mathematics of probability.* [Thurston 1992, p. 649]

In [Wood 1990], another direct comparison is made between the fuzzy calculus and probability as a means of representing imprecision in engineering design. For a particular function, the input parameters are first represented as fuzzy numbers and the output (a fuzzy set membership function) is calculated using the extension principle. Then the input parameters are represented as probability density functions (PDFs) and the output (a probability density function) is calculated using the usual axioms of probability theory. The PDF result is normalised to take a maximum value of 1 so that it is directly comparable to the fuzzy set membership function and the two are compared. This is repeated for several functions of increasing complexity.

There are two major differences between the fuzzy and the probabilistic results. The fuzzy result always has its peak at the point corresponding to the peaks of the input preference functions, whereas this is not generally the case for PDFs. The second major difference between the fuzzy and probabilistic results is that the fuzzy results are generally "broader" (i.e. with greater preference values for points towards the extremes of the support for the result) than their probabilistic counterparts. It is also explained in the section entitled "Extension Principle" in Appendix B that the fuzzy calculus displays a property which Wood and Antonsson term the *backward path* - the property that the preference values of the input parameters can be deduced from the preference value of the output parameter. Wood and Antonsson conclude that the differing peak position and the fact that the probabilistic approach doesn't provide the *backward path* described above render probability theory inferior to the fuzzy approach for modelling design imprecision.

Wood and Antonsson interpret the fuzzy set membership function as a possibility distribution (in the sense that the support is the range of *possible* values) but with its peak corresponding to the *preferred* value. If one accepts this interpretation, then the fact that the peak of the output parameter corresponds to all the input parameters taking their preferred values represents the statement that "the preferred value of the output parameter is that which results from the preferred values of each input parameter". It is important to note that this interpretation does not reflect the *likelihood* of a parameter taking a particular value, due to the designer's preferences. With such an interpretation (which would be required for example for risk assessment), both the position of the peak of the output parameter and the broad shape of the distribution obtained using the fuzzy approach are incorrect.

We also observe that with the fuzzy approach the width (support) of the membership function of the output parameter represents the total range of *possible* values (rather than *probable* values). It is difficult to think of a useful interpretation for this width which is not related to likelihood, and thus invalid. The usual probabilistic interpretation of the "total amount of uncertainty" in the model is useful because it tells us the range of likely final output values. Again, such an interpretation of the fuzzy distribution support would be an overestimate and would be incorrect.

In choosing the representation for uncertain numerical information in the risk model, the most fundamental requirement was that the risk model could be evaluated to produce values representing likelihood as opposed to possibility. Therefore, it was concluded that uncertain numerical attribute values would be represented by probability distributions, even when, as in the case of explicit design attributes under the control of the designer, the uncertainty concerned degree of belief. This choice is justified by the adoption of the personalist view of probability.

5.2.2 Representing Categorical Uncertainty

In Chapter 4 the risk model was shown to be required to represent three basic types of categorical uncertainty; firstly, uncertainty due to the diversity of auxiliary models which may be available to evaluate implicit attribute values; secondly, uncertainty concerning events in the environmental model (which objects or attribute values will appear in the design environment when the design is completed) and thirdly, uncertainty concerning events in the design model (design choices between alternatives which have not yet been made).

In Section A.5.4 of the requirements document in Appendix A it is noted that the industrial partners were concerned that combination of evidence from different auxiliary models would give rise to results which would be difficult to audit or justify. It was also found that a set of auxiliary models for a given implicit attribute value could often be ordered into a list such that each model in the list required the results of its predecessor plus some additional inputs and generated more certain results. In [Mileham 1993] for example, a parametric cost estimation system for injection moulded components is described which is based on a hierarchy of estimating equations. The first equation is a function solely of weight, the second is a function of weight and production volume and so on. Thus a cost estimate can initially be made on weight alone, with a more accurate estimate available when production volume is also known and so on. For these reasons, it was decided to represent the diversity of auxiliary models simply as a set, to be used one at a time, in turn as the necessary information became available. Therefore an explicit representation for the uncertainty due to this diversity was not required. Note however that it was required to represent explicitly the numerical uncertainty introduced by the use of a particular auxiliary model, and that this was represented by the probability distribution function and parameters generated by the heuristic method which defined the auxiliary model.

Categorical uncertainty concerning events in the environmental model differs conceptually from categorical uncertainty in the explicit design model because the latter is largely (though not entirely) under the control of the designer and thus almost always concerns degree of belief, whereas the former may concern either degree of belief or frequency of occurrence. The personalist view of probability however allows us to use probability theory to represent both of these cases and to combine them into a single model which generates results describing likelihood, as is required of a risk model. The representation chosen for both types of categorical uncertainty for independent events was therefore a set of alternative objects in the risk model, each of which is assigned a probability value which can be interpreted as the likelihood that the object will be included in a particular instance of usage of the final design. It is possible to include sets of alternatives without including probability values, but they are not included in the evaluation of the risk model. Where the uncertain event (the choice or occurrence of an alternative) is dependent on other random variables in the

model, the set of alternatives are included in the risk model but there is no explicit representation for the uncertainty since the choice between alternatives is made by evaluating some deterministic expression of the other random variables. It was decided that statistical dependence between the choice of alternatives and other random variables would not be explicitly modelled (e.g. using correlation coefficients) because it was felt that the definition of such dependencies would be too complex and distracting for the intended users and situation of use of the risk tool. In practice such correlations usually arise from hidden causal dependency networks and the inclusion of the causal network in the risk model would improve its legibility to other members of the design team.

5.3 References

[Ayer 1973], Ayer, A. J., "*The central questions of philosophy*", pp. 163-183, Harmondsworth, Middlesex, England: Penguin, 1976.

[Carnahan 1994] Carnahan, J. V., Thurston, D. L., and Liu, T., "Fuzzy Ratings for Multiattribute Design Decision-Making", *Transactions of ASME- Journal of Mechanical Design*, vol. 116, pp. 511-521, June. 1994.

[Choobineh 1992] Choobineh, F. and Behrens, A., "Use of Intervals and Possibility Distributions in Economic Analysis", *Journal of the Operational Research Society*, vol. 43, no. 9, pp. 907-918, 1992.

[Cui 1990] Cui, W. and Blockley, D. I., "Interval Probability Theory for Evidential Support", *International Journal of Intelligient Systems*, vol. 5, pp. 183-192, 1990.

[Díaz 1989] Díaz, A. R., "A strategy for optimal design of hierarchical systems using fuzzy sets" in *The 1989 NSF Engineering Design Research Conference*: At College of Engineering, University of Massachusetts, Amherst, J. R. Dixon Ed., pp. 537-547, June 1989.

[Dubois 1980] Dubois, D. and Prade, H., "*Fuzzy sets and systems: theory and applications*", New York: Academic Press, 1980.

[Evans 1989] Evans, G. W., Karwowski, W. and Wilhelm, M. R., "An introduction to fuzzy set methodologies for industrial and systems engineering" in *Applications of Fuzzy Set Methodologies in Industrial Engineering*. Amsterdam: Elsevier, 1989.

[Freund 1980] Freund, J. E. and Walpole, R. E., "*Mathematical Statistics*", London : Prentice-Hall International, 1980.

[Keynes 1921] Keynes, J. M., "Historical retrospect" in *A Treatise on Probability*. London: Macmillan and Co Ltd, pp. 79-91, 1921.

[Mileham 1993] Mileham, A. R., Currie, G. C., Miles, A. W. and Bradford, D. T., "A parametric approach to cost estimating at the conceptual stage of design", *Journal of Engineering Design*, vol. 4, no. 2, pp. 117-125, 1993.

[Moore 1966] Moore, R. E., "*Interval analysis*".Prentice-Hall: Englewood Cliffs, NJ, 1966.

[Otto 1991] Otto, K. N. and Antonsson, E. K., "Trade-off strategies in engineering design", *Research in Engineering Design*, vol. 3, pp. 87-103, 1991.

[Otto 1993] Otto, K. N. and Antonsson, E. K., "Extensions to the Taguchi method of product design", *Journal of Mechanical Design*, vol. 115, pp. 5-13, March. 1993.

[Otto 1994] Otto, K. N. and Antonsson, E. K., "Design parameter selection in the presence of noise", *Research in Engineering Design*, vol. 6, pp. 234-246, 1994.

[Skidelsky 1992] Skidelsky, R., "*John Maynard Keynes - the economist as saviour, 1920-1937*". London: Macmillan, 1992.

[Thurston 1992] Thurston, D. L. and Carnahan, J. V., "Fuzzy Ratings and Utility Analysis in Preliminary Design Evaluation of Multiple Attributes", *Transactions of the ASME: Journal of Mechanical Design*, vol. 114, pp. 648-658, Dec. 1992.

[Wood 1989] Wood, K. and Antonsson, E., "Computations with imprecise parameters in engineering design: background and theory", *Transactions of the ASME: Journal of Mechanisms, Transmissions and Automation in Design*, vol. 111, pp. 616-625, Dec. 1989.

[Wood 1990] Wood, K., Antonsson, E. and Beck, J., "Representing imprecision in engineering design: comparing fuzzy and probability calculus", *Research in Engineering Design*, vol. 1, pp. 187-203, 1990.

[Zadeh 1965] Zadeh, L. A., "Fuzzy Sets and Systems" in *System Theory* (J. Fox ed), New York: Polytechnic Press, 1965.

[Zadeh 1975] Zadeh, L. A., "The concept of linguistic variable and its application to approximate reasoning", *Information Science*, vol. 8, 1975.

[Zadeh 1978] Zadeh, L. A., "Fuzzy sets as a basis for a theory of possibility", *Fuzzy Sets and Systems*, vol. 1, pp. 3-28, 1978.

CHAPTER 6

Object-orientation

In this chapter, the object-oriented paradigm is explored and the reasons for choosing this paradigm for the risk model are presented. In the first section, an overview of the object-oriented (OO) philosophy is presented and the ideas of OO programming, OO modelling, OO analysis and design and OO database management systems are introduced. The second section contains an overview of the Fusion methodology. The choice of the Fusion methodology as an analysis, design and implementation tool for development of the risk toolkit and as a modelling notation for the risk model is justified. In the third section, an overview of some of the language features of C++ is presented, and its choice as an implementation language is explained. Finally, in the last section, the design decisions concerning the risk toolkit which have been discussed in Chapter 6 are summarised.

6.1 Introduction to Object-Orientation

6.1.1 Principles of Object-Orientation

Object-orientation is an approach to modelling and programming which is based on the simple and intuitively attractive concept of a 1:1 relationship between components of the model or computer program (objects) and entities in the real world. The real world entities may be physical, or they may be more abstract concepts - but they are the natural entities in terms of which the user of the model or program thinks about the domain being modelled. As Stroustrup points out [Stroustrup 1991], most object-oriented (OO) computer programs will also include a second kind of object which are the objects required to provide the program functionality; these are in 1:1 relationship with the natural entities in terms of which the programmer thinks about or describes the implementation.

In traditional procedural computer programs or models, data is moved through a series of procedures which act upon it, as illustrated in a Data Flow Diagram. As the model becomes large and complicated it becomes increasingly difficult for the modeller to envisage the effect of a change to the data or procedures. An object-oriented model, on the other hand, consists of a set of autonomous objects, each containing their own data, which communicate by a tightly defined interface (this communication is sometimes characterised as message passing). When an object receives a message, it acts upon its own data using its own procedures (often referred to as methods). This *procedural and data abstraction*, where both the data and the methods (or behaviour) which characterise a particular type (or *class*) of object are identified, is one of the four defining concepts of object-orientation. The other three are *encapsulation*, *inheritance* and *polymorphism*.

Encapsulation is the hiding of private data and methods within an object, leaving only public properties visible. This makes modifying the data or methods of a class of objects which is already in use in a program or model much safer; the programmer is free to modify private data and methods, for example to fix a problem or improve performance, knowing that the only changes this will cause in the remainder of the program can be checked by examining the effect on the public data and methods of this particular class. Thus the combination of abstraction and encapsulation localises the effect of changes to data and methods, allowing larger and more complicated models to be built, modified and maintained than with the traditional procedural approach.

Inheritance is used to create *class hierarchies*. A *class* is a categorisation, or classification, for a set of similar objects - a set of objects which differ only in their data values are said to belong to the same class. Classes are arranged in an inheritance hierarchy, where a child class may inherit both data and methods from its parent class or classes (classes above it in the hierarchy) but will also have its own, specialised, data and methods. An inheritance relationship between two classes, (e.g. in a model of a car, the class AlloyWheel is the child of the class Wheel) may be interpreted as an "is a" relationship in natural language (e.g. "an alloy wheel is a wheel"). One of the major benefits of inheritance is the opportunities offered for re-use of existing classes in new contexts. If a requirement arises for modelling an aluminium wheel, it can inherit from the existing Wheel class. Or, if a large alloy wheel is required, it could inherit from AlloyWheel. The other important mechanism for re-use is containment (or aggregation) which is represented in natural language as a "has a" relationship (e.g. "a car has four wheels"). If a model of a lorry is required at a later date, the new Lorry class can contain, and hence re-use, the existing Wheel class.

Polymorphism is the property of an object responding appropriately, according to its class, to a standard message (or method invocation). There are many examples of abstract concepts which could usefully be represented as standard messages, aiding clarity and legibility: for example, multiplication. A message sent to an integer telling it to multiply itself by itself, should perform an integer multiplication. In a polymorphic implementation, the same message when received by a matrix, would perform a matrix multiplication, and so on. Polymorphism becomes particularly useful in OO programming when combined with *late-binding*, where the exact class of an object is not known until run-time - the same piece of code can be invoked on different classes of object and each will respond appropriately. For example, consider a program to calculate the total area of a set of different shapes, selected at run-time from a pick-list by the user; the high level code sends a standard message to each shape instructing it to return its area, and simply adds the results. This usually reduces the total amount of code required and also aids legibility because the high-level code is not cluttered with details concerning how to calculate the area of a square, a triangle and so on. Most importantly, if a new shape class is added, the high level code requires no alteration.

It could be argued that some of the principles of OO programming (OOP) can be adhered to without using an OO language - C programmers for example, try to achieve data and method abstraction by collecting related variables and functions in small modular implementation files, to encapsulate private data and functions within the scope of the implementation file and to achieve a limited form of polymorphism simply through consistent function-naming. More sophisticated techniques using call-back functions and features of the

operating system to allow the dynamic addition of new program components at run-time are also used. The principles of OOP are an extrapolation of "good programming practice", not a revolution.

The main reasons for choosing an OO paradigm for the risk model were three-fold. The first, identified in Chapter 4 when the requirements were discussed, was the need to facilitate rapid model-building. If the time and effort required to build the risk model detracts from the designer's main task of actually designing, then the risk tool will not be used. The provision of ready-made, pre-defined object classes frees the designer from much of the detail in the model and simplifies model building. If the object class definitions include historical data, and methods to utilise it, then this too will simplify the designer's task. Knowledge, encapsulated in object classes, can be re-used either by component re-use, or by inheritance. The second reason was the object-based, constructive and incremental nature of the design process; designers build complex systems by combining simple components, often in a hierarchical fashion. An OO model can support this process, allowing objects to be added to the model as detail is added to the design. The third reason was the need to represent the concept of abstraction and to support a design process of refinement which includes moving from abstract to specific information. Some of these reasons have motivated the development of other OO product data models for use in early design (see for example, [Murdoch 1994]).

6.1.2 History of Object-Orientation

OOP began its history in the 1960s with the development of Simula67, a discrete event simulation language. Smalltalk, developed at Xerox Palo Alto Research Centre (Xerox PARC) in the 1970s, is still considered by many to be the OO programming language which is most true to the principles of object-orientation, but has never really been widely adopted in the commercial programming community. Xerox were the originators of the graphical user interface (GUI) and Smalltalk was the first language to adopt the now universal paradigm of representing GUI entities such as windows, icons and menus as objects.

Work in the area of knowledge-representation and models of human reasoning has been highly influential in the development of OO. Frames, proposed in 1975 by Marvin Minsky, and their less structured and hierarchical predecessor, semantic nets, are knowledge representation systems based on networks of objects with procedural and data abstraction, inheritance and also default data values. The Knowledge Representation Language (KRL), developed by Bobrow and Winograd in the late 1970s at Xerox PARC, was an attempt to integrate frames into a practical language.

Over the next 20 years, as GUIs became widespread and software systems became larger and more complex and the principles of OO became more necessary to handle this complexity, OOP gradually moved into the mainstream of computing practice; OO extensions to already widely used languages eased the process. Bjarne Stroustrup of AT&T laboratories designed and implemented C++ (a superset of C [Stroustrup 1991]) and its first version was released in 1983. Although criticised by OO purists as compromising the principles of OO due to its origins in C (for example, the syntax distinguishes between base-typed variables such as characters, floating point values etc. and objects) and also for its provision of "dangerous" (but fast) features such as direct access to pointers (object addresses) and explicit casts (where the programmer can freely change the type/class of a variable/object), it is probably the most widely used OO language today. In 1986 the first version of Eiffel was released [Meyer 1989][Meyer 1992], a pure OO language developed by

Bertrand Meyer at ISE (interestingly, Eiffel is compiled into bytecode and then into C and thus can easily be integrated with software written in C++). Recently, Java has become very significant in the OOP community. In 1995 the first version of Java was released by Sun Microsystems; Java is, arguably, a pure OO language with a similar but simpler syntax to C++, but with safety features such as no pointers and automatic garbage collection. Java programs are composed of software components which can be distributed anywhere on the World Wide Web (WWW), and can be dynamically extended by the addition of new components. Complete platform independence and the ability to easily write distributed programs, achieved via the WWW, are probably Java's most innovative features.

6.1.3 When to Inherit?

The inheritance mechanism is one of the most powerful features of OO programming and modelling languages, however it is not a panacea and when used inappropriately can cause more problems than it solves. Two interrelated issues in particular are controversial in almost all OO development. Firstly, when to use inheritance in order to re-use information and when to use aggregation (or containment). And secondly, whether or not to use multiple inheritance (when a child class may directly inherit from more than one parent class) and how to cope with the implementation difficulties which this entails. A single-inheritance hierarchy was chosen for the risk model (see Chapter 10), and the issues behind this decision are discussed here, as well as important general issues relating to the use of inheritance in OO models and programs.

The two issues are related because some languages (Smalltalk for example) only permit single inheritance and in practice this can lead to the use of containment in situations where multiple inheritance would perhaps be more intuitively natural. On the other hand, languages which permit multiple inheritance (such as Eiffel and C++), need to include more complex language features to cope with the situation where a child class inherits the same data or method indirectly from two parent classes. This will occur when the two parent classes (say A and B) themselves have a common parent; thus the child inherits two copies of the data defined in the common parent, leading to ambiguity, and if the same method has been implemented differently in A and B then it will also be ambiguous which of the two versions of the method should be used when it is invoked on the child. The two other difficulties with multiple inheritance are the (generally small) additional time and memory requirements involved in resolving references to methods and data in multiple parent classes and, more importantly, the great care needed to avoid excessively complex inheritance networks which are very difficult to understand and maintain.

In [Armstrong 1994] the authors argue that the concept of inheritance loses expressive power because of the many different ways in which it is used in OO programming and modelling. They compare the current use of inheritance in OO with the use of the GOTO statement before the development of more specialised constructs with more expressive power such as IF THEN, REPEAT UNTIL etc. They note that situations where code can be re-used through inheritance do not always coincide with situations which can be modelled using the "is a" relationship. They also note that a distinction can be drawn between the relation where a child class shares the attributes of its parent (an *attribute* "is a") and the relation where the members of the child class form a subset of the members of the parent class (a *subset* "is a"); for example, moveable points share attributes with points, but are not necessarily a subset of the set of points. The simplest interpretation of inheritance is that both such relations exist. Armstrong and Mitchell also point out that a taxonomy based on

behaviour may well have a different hierarchy from one based on structure; for example, from a behavioural point of view, helicopters, hot air balloons and aeroplanes can all be regarded as aircraft, but they do not inherit any structure from aircraft. Moving from modelling to programming, Armstrong and Mitchell identify several uses to which inheritance mechanisms are put in OOP and argue that only those which correspond to behavioural "is a" relationships should be used, in order that sub-classes can safely be substituted for super-classes. They argue that the use of inheritance to implement heterogeneous collections of objects which are not related to a common parent by an "is a" is unnecessary and leads to flawed hierarchies. The use of inheritance in "programming by contract" (where the subclass must fulfil the contract of its super-class, and so a sub-class can always be safely substituted for its superclass) corresponds to a behavioural "is a" relation and is thus viewed as legitimate. Armstrong and Mitchell argue that inheritance should only be used as a version control mechanism, to localise or propagate changes, as the programmer requires, when a behavioural "is a" exists. They argue that haphazard use of inheritance simply to maximise code re-use, and the use of inheritance in place of aggregation to model "is a part of" relations, are abuses which lead to flawed hierarchies.

The approach to inheritance taken in implementing the risk toolkit and also in building the OO risk models was as close to that advocated by Armstrong and Mitchell as was practically possible.

6.1.4 Object-Oriented Analysis and Design

As OOP has become more widespread as an implementation tool, the need has arisen for supporting methods and tools to enable the developer to model the problem domain and to design the program in a way which is consistent with the OO approach. This allows continuity between the analysis, design and implementation phases and provides clear documentation of design intent. OO methodologies such as the Object Modelling Technique (OMT), developed by Rumbaugh [Rumbaugh 1991]), Booch [Booch 1991] and Coad & Yourdon [Coad 1991] prescribe an OO analysis and design (OOAD) process and define sets of graphical and structured text models which can be used to record and communicate the developer's views of the problem domain and design. Some OOAD methodologies are *evolutionary* (e.g. OMT) in that they are based on pre-existing structured programming methods such as data flow diagrams, and others are *revolutionary* (e.g. Booch) in that they differ significantly from traditional methods and are entirely based on the OO paradigm, for example including dynamic models of object creation, deletion and interaction. Revolutionary methods are more closely allied to OO programming languages such as C++ and provide more support for their language features. The proliferation of methodologies (22 were compared in [Dock 1992] for example) has lead to attempts at standardisation.

The Fusion methodology which was used in this research (see Section 6.2) is a synthesis of pre-existing OOAD methods - in particular OMT and Booch - and a technique called CRC (Class Responsibility Collaborator). CRC is an exploratory technique for use by teams of developers, where each class is given an index card on which its responsibilities (functional requirements) are recorded and then its collaborators (other classes which would need to be consulted to meet its responsibilities) are identified. It has been debated in specialist forums on the Internet whether Fusion is evolutionary or revolutionary; it could be argued that its support for dynamic object modelling and exclusion of data flow diagrams make it revolutionary, but it must be acknowledged that the lack of support for abstract and parameterised classes is a

weakness. Very recently Rumbaugh and Booch have joined forces to begin to define a Unified Modelling Language, and the developers of Fusion have also agreed to adopt this standard.

Computer Aided Software Engineering (CASE) tools have been developed to support such OO analysis and design methodologies. At their most basic, CASE tools are little more than drawing packages which have been specialised for a particular methodology. However, far more sophisticated tools are now available which enable many automated consistency checks to be made between different views and which can, to a limited extent, automatically generate code from the graphical and textual views built by the developer/s. CASE tools are also available which specifically address the difficulties of team-based development, providing version control and traceability, and addressing issues such as ownership. In this research, the FusionCASE CASE tool from SoftCASE consulting was used in the development of the risk toolkit and was also integrated into the development of risk models (see Section 6.4). The version of FusionCASE which was used in this research (Version 1.4) is essentially a single-user system which provides a graphical editor for the Fusion graphical views and a structured text-based editor, which imposes the correct syntax, for the text-based Fusion views. Quality reports can be automatically generated which identify many of the inconsistencies which may arise in a Fusion model, and C++ class headers can also be automatically generated (although there appear to be some implementation difficulties with this last feature).

6.1.5 Object-Based Information Models

Object-based information models are now widely used; in [Eastman 1994] a comparison between several information models used for product design is presented including IDEF1X, NIAM, Express and EDM (a product modelling language developed by Eastman, Chase and others). Express [Schenk 1994] is an "object flavoured" information modelling language which was developed to fulfil the modelling needs of the International Standard Organisation (ISO) STEP project. The STEP standard is the ISO 10303 international standard for the exchange of product model data. A STEP Express data model (or schema) for the data used by an information system allows instances of that data to be exchanged with other information systems - provided that the Express models used by both systems comply with the STEP standard for their particular domain. The STEP standard specifies data models in Express for many types of information which may constitute part of a product definition: for example it includes representations for finite element analysis (FEA) data, product structure configuration, shape tolerances, materials, etc. As it is so widely used, a brief description of the Express modelling language, and its partner graphical modelling language, Express-G, is given here as an illustration of object-based data modelling and is contrasted with C++ (see Section 6.3). An instance language, Express-I, also exists and can be used to build test harnesses for Express information models.

An Express schema defines a set of *entity types*, or classes, defines their inheritance hierarchy, their *attribute* names and types (which are binary, integer, real, string or Boolean) and their *relationship* names and classes (links or pointers to other entities). Collections of entities or base-typed values of the following kinds may be defined: *arrays* are ordered with fixed size, *lists* are ordered with variable size, *sets* are un-ordered with no duplication permitted and *bags* are unordered with duplication permitted.

An example Express schema to model cars and motorcycles might contain the following definitions (some features of the syntax are explained below):

```
ENTITY vehicle
 SUPERTYPE OF (ONEOF(car, motorcycle));
 has_engine : engine;
UNIQUE
 has_name : name;
END_ENTITY

ENTITY car
 SUBTYPE OF (vehicle);
 has_wheels : ARRAY[5:5] OF wheel;
END_ENTITY

ENTITY motorcyle
 SUBTYPE OF (vehicle);
 has_wheels : ARRAY[2:2] OF wheel;
END_ENTITY

ENTITY engine
END_ENTITY

ENTITY wheel
END_ENTITY
```

This example schema would be represented graphically in Express-G as shown in Figure 6-1.

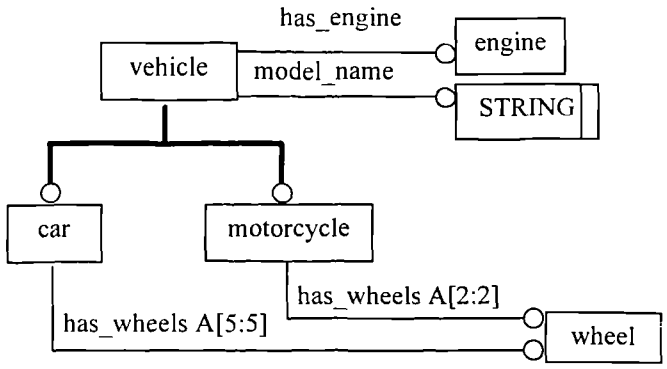


Figure 6-1: Example STEP Express schema represented with Express-G

Unlike C++ and Smalltalk, for example, Express does not require that two classes which inherit from a common superclass should be disjoint. This is possible because in the Express instance language, Express-I, the programmer may specify which super-class instances to create individually, whereas in C++ the programmer specifies a single class to create and the compiler automatically creates the necessary super-class instances. Thus in C++ the programmer may create an instance either of `vehicle`, or of `car`, or of `motorcycle`. In the latter two cases, the creation of the `vehicle` data is implicit and will be automatically handled by the compiler. (To create an object which is both a `motorcycle` and a `car` in C++, the programmer must create a new class which multiply inherits from both, and then create an instance of this new class). In Express-I, the modeller may choose to explicitly create, and associate with one-another, any combination of `vehicle`, `car` and `motorcycle` objects, subject to the kind of inheritance which is specified in the schema. If `vehicle` is defined to be an ABSTRACT class in the schema then it may only be instantiated when associated with a `car` or a `motorcycle` (C++ has a similar language construct). The modeller may also specify that the `vehicle` superclass has subclasses which are ONEOF(`car`, `motorcycle`) or ANDOR(`car`, `motorcycle`) or AND(`car`, `motorcycle`). ONEOF(`car`,

motorcycle), as above, means that the intersection of cars and motorcycles is empty - that a vehicle instance may only be accompanied by one of a car instance or a motorcycle instance. ANDOR(car, motorcycle) means that a vehicle instance may be accompanied by one or the other or both. AND(car, motorcycle) means that a vehicle instance must be accompanied by both.

In Express, the entity class definition also defines the *constraints*. Three types of constraints can be defined; *uniqueness constraints*, *local constraints* and *existence constraints*. *Uniqueness* constraints (indicated by the UNIQUE keyword) are used to represent the situation where the value of an attribute or relationship is constrained to be unique to each instance of the entity. Thus in the example, every instance of vehicle must have a unique name. *Local constraints* are logical expressions - if a local constraint evaluates to FALSE then the entity instance is invalid. In their simplest form local constraints are used to indicate permissible ranges of attribute values, but the Express language also allows algorithms to be defined to fully represent complex constraints. *Existence constraints* are used to indicate that an entity instance is only valid if another instance of a related class exists. There are no explicit language constructs in C++ for representing uniqueness and local constraints but existence constraints can be explicitly represented using C++ *references* (see Section 6.3).

Express also permits modelling of *derived attributes*; these are attributes whose values can be calculated from other attribute values in the model. For example, all cars of a particular model (e.g. Rover 2000) will have the same manufacturer - thus, in the example below, the manufacturer for a car (given by attribute is_made_by) can be derived from the manufacturer of its model (given by has_model_type.is_made_by):

```
ENTITY car;
 has_model_type : car_model;
   ...
DERIVE
 is_made_by : manufacturer := has_model_type.is_made_by;
END_ENTITY
```

In this research, it was decided that the inclusion of support for uncertainty in the OO information model should be transparent to the user. The schema for the risk model (the class definitions) built by the user should model only the design domain and should not need to include, for example, probability distribution functions, or other information related to how uncertainty was represented. This placed constraints on the modelling language which could be used; in particular, it needed to be sufficiently simple that uncertainty could be incorporated at run-time into any relevant aspect of the model - for example uncertain object pointers or uncertain attribute values (Chapter 8 contains an overview of the complete modelling methodology adopted). There was no clear means available for transparent incorporation of uncertainty into Express-style constraints. Also the mapping between Express and C++ is not direct.

For all these reasons, it was decided not to use Express for the OO information model. The Fusion method provides a graphical and text-based notation for defining object classes, and it was decided that this

representation should be adopted for the risk model schema. Thus, the user-defined risk model classes could be developed graphically using the Fusion CASE tool.

6.1.6 OO DBMS

An information model, for example represented in Express, specifies the information which should be stored. It does not specify *how* the information should be stored. In most applications, the information model will be stored in a database of some sort. Currently, the majority of database applications still use a relational database management system (DBMS) but, despite still being relatively new technology, object-oriented database management systems (OO DBMS) are gradually gaining wider acceptance. An OO DBMS stores the data for each object in a single place, indexed by a unique object identifier (object ID), rather than scattering the data between different tables as in traditional relational systems. Relationships (links) between objects are usually represented directly using the object ID and thus object links are followed, not by performing "joins" (in which the records from two tables are combined by associating a field in the first table with a field of the same data type in the second), but by using the object IDs directly. An OO DBMS will usually provide support for run-time schema querying - where applications can query meta-data about the objects, for example the class name, the names and types of the class attributes and methods, the name of the class, the super and sub-classes etc.

Hybrid OO-relational products exist which enable object representations to be made in relational databases. Truly OO DBMS are less widely used, and they usually allow methods to be stored in the database schema as well as data structures - those which do not are termed *passive*. Truly OO DBMS systems offer considerable advantages of speed over hybrid systems when following object links. Most OO DBMS also provide some support for object versioning. Three OO DBMS products were considered in particular for use in this research; Ontos (from Ontologic), O2 (from O2 Technology) and ObjectStore (from Object Design). The brief comments which follow concern the state of the products at the time that the survey was carried out (early 1994).

All three provided object persistence, an interface to C++ and an object browser for examining persistent objects. **O2** included a graphical user interface (GUI) building tool - this was an integrated environment for developing applications which store their data in O2. An OO extension to SQL, called O2 SQL, was also provided, for performing queries on the DB. Methods were stored in the DB schema as well as data structures, and it was possible to modify the schema (the class definitions) when there were existing instances of the classes, and O2 would automatically perform the necessary changes to the existing objects (this is termed *schema evolution*). **Ontos** provided a graphical schema designer but it was not possible to invoke methods or run queries from it, and thus other tools would have been required to build a GUI for applications whose data was stored in Ontos. Also, at the time, Ontos did not support schema evolution. **ObjectStore** was a passive OODB which did not include method storage, and used a different persistence mechanism from other systems, based not on object IDs but on storing an exact image of memory to disk and then, when re-loading from disk, "swizzling" the pointers from one object to another so that they point to the correct place in memory. One advantage of this approach is that the choice of compiler, and even language, is in principle up to the user. No proprietary compilers or GUI builders were provided. Once an object is loaded from disk, access is very fast - object references are ordinary C++ pointers. The objects are small

because they are not inherited from a "Persistent" base class. The first load of an object from disk is slow however. Exactly the same C++ code can be used to create transient or persistent objects. ObjectStore provided support for limited schema evolution.

The dual objections to the use of an OO DBMS in this research were the initial financial outlay required and the perception, particularly among the industrial collaborators, that the technology was not yet mature. The costs, which were typically several thousand pounds for non-academic licenses, would have precluded carrying out case studies in the industrial partner companies. It was decided therefore to provide object persistence and simple run-time schema querying by writing C++ code from scratch. However, the format chosen for the persistent objects was designed to support possible migration to an OO DBMS in the future.

6.2 The Fusion Method

The Fusion method [Coleman 1994] is a methodology for the analysis, design and implementation of object-oriented systems. The method defines several types of diagram and text-based schema which may be used to model the problem domain, design the solution and begin to produce an implementation. These diagrams and schemata represent differing "views" of the system under design and the power of the Fusion method is that consistency checks may be performed between these differing views or models. Many of these checks are automated in the FusionCASE CASE tool.

The views are grouped into those which are part of the analysis stage and those which are part of the design stage. It is intended that the steps taken to generate the views within each stage should be iterative. Figure 6-2 shows the six main views provided (shown heavily boxed in the figure), with the interface model itself containing an additional three different views. In addition to these nine views of the design, a data-dictionary is also constructed which contains a description of all of the entities in all of the views. There follows a brief description of each view:

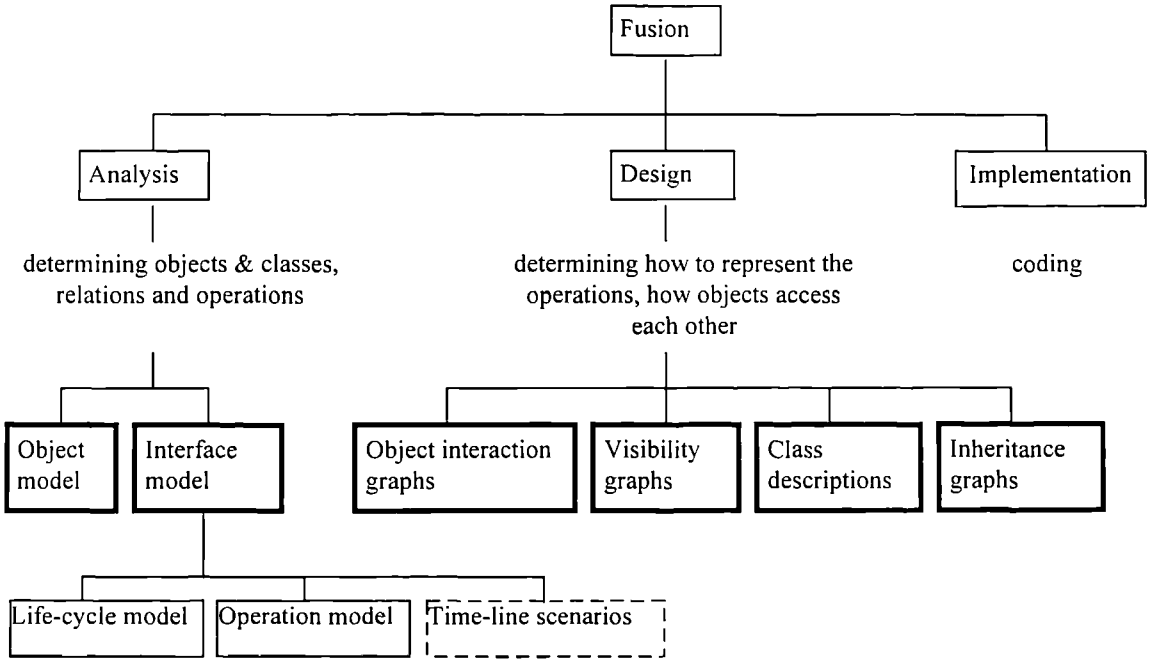


Figure 6-2: The views provided by the Fusion methodology

6.2.1 Analysis: Object Model

The object model is essentially an entity-relationship diagram describing the problem domain, augmented with inheritance. The entities are classes which may have attributes (the relationships may also have attributes). In Fusion (like C++), a distinction is made between objects (which have class) and values (which have base-type), and attributes may not be object-valued. Such a relationship between classes should instead be represented in the object model by either a relationship between the classes or by aggregating one class inside the other. The base types are stored in the data dictionary and may be defined by the modeller to suit the particular programming language or environment that will be used. One limitation of the Fusion methodology is that no provision is made for modelling aggregates (or collections) of base-types - the aggregates must themselves be defined as types in the data dictionary. For example, the data dictionary might include the following two base-type entries:

Name :	FLOAT
Description :	C++ double length floating point number. In the range 1.7E +/- 308 (15 digits).

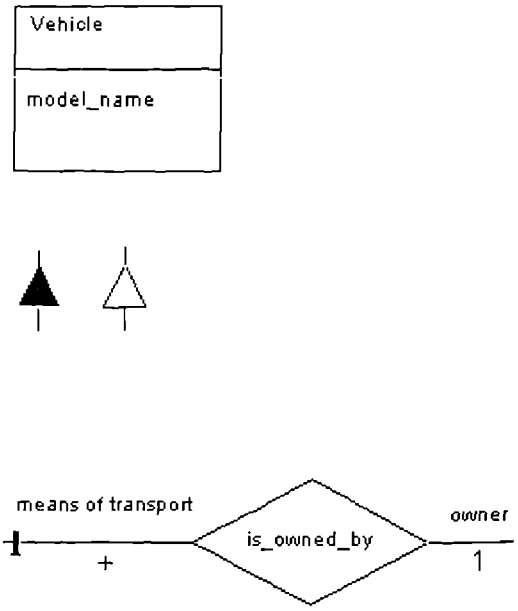
Name :	FLOAT[ ]
Description :	variable cardinality FLOAT

This defines the base-types "FLOAT" and "FLOAT[ ]". The notation "FLOAT[ ]" has been invented to represent a list of FLOATSs of variable length. The notation "FLOAT" is not raw C++ - it is assumed that a type definition for "FLOAT" will eventually be implemented in C++. If C++ template collection classes, such as List and Array, are available then the notation "List<FLOAT>", for example, might be used instead. The Fusion methodology does not specify the base-types which may be used. The data dictionary also contains a record of the base-type of each attribute of each class:

AttributeName :	average_weekly_mileage
AttributeType :	FLOAT

The graphical notation for the Object Model is illustrated in Figure 6-3 and Figure 6-4. Notice that one person may own one or more vehicles (indicated by the cardinalities marked on the relationship), and that whilst all vehicles must have an owner (indicated by the totality marker), it is possible that a person will not own a vehicle. Also, in Figure 6-4, the use of a solid triangle implies that a vehicle cannot be both a car and a motorcycle and also that no other types of vehicle exist in the problem domain. We have excluded the possibility of including lorries in the model.

Having captured the key entities and relationships in the problem domain and represented them in the object model, the system boundary may be defined. The system boundary can be drawn onto the object model, identifying which classes and relationships are part of the system under design and which are external. Those classes which lie outside the system boundary are identified as agents - agents are entities which interact with the system under design but are not a part of it.

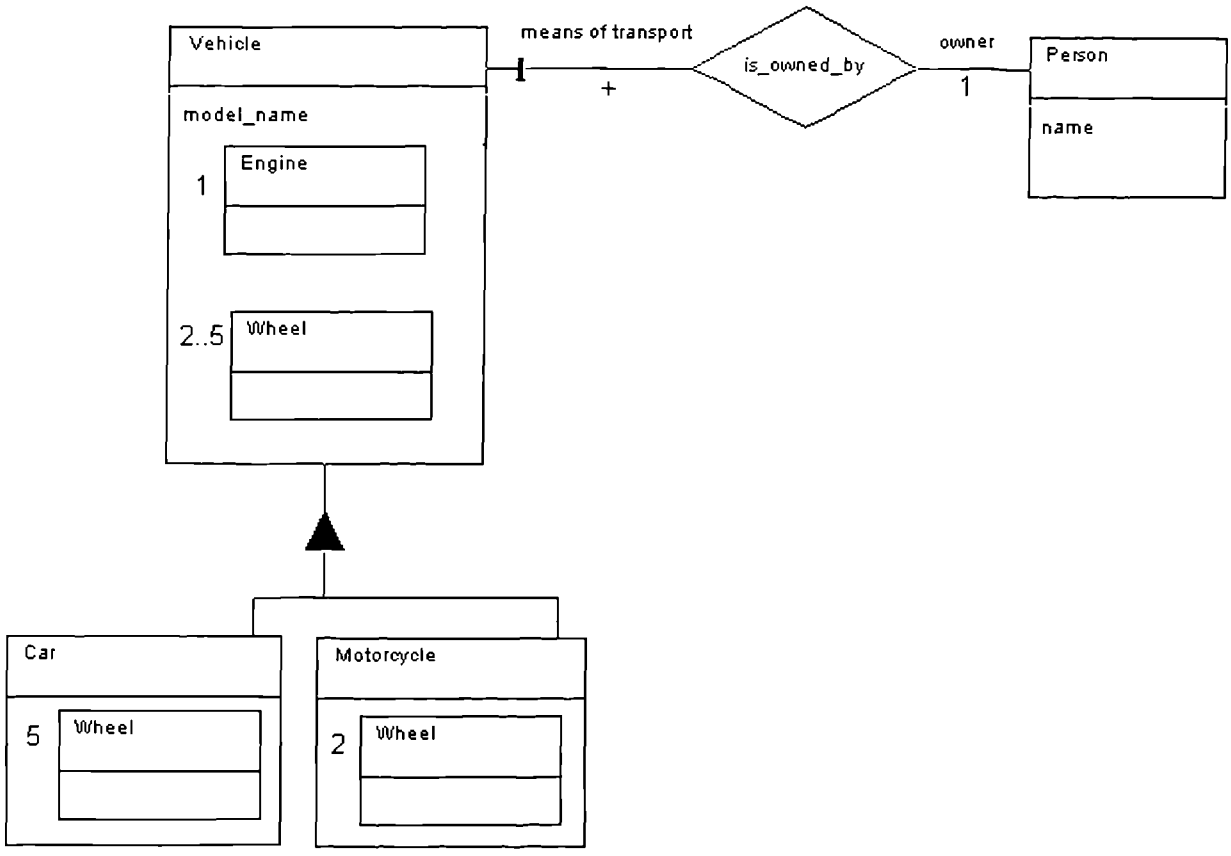


Rectangles represent **classes**. `Vehicle` is the class name. Objects of class `Vehicle` have an attribute called `model_name`.

Triangles represent **inheritance** relationships between classes. Solid triangles are used when the sub-classes are disjoint and complete, otherwise hollow triangles are used.

Diamonds represent other **relationships** between classes. The name of the relationship is `is_owned_by`. The **role** of each class appears above the arc, and the **cardinality** appears below. A cardinality of * denotes zero or more, + denotes 1 or more, and [N..M] indicates a range from N to M. The **totality marker** on the left indicates that all instances of the class take part in such a relationship.

Figure 6-3: Some of the Fusion object model notation



Aggregation is represented by placing the child class within the parent class. The **cardinality** of the aggregation appears to the left of the child class. Thus an instance of class `Car` contains exactly 5 instances of class `Wheel`.

Figure 6-4:Fusion object model

Suppose that the example shown in Figure 6-4 was part of the analysis for a program to store and retrieve descriptions of the vehicles owned by people; the program might be used by a company which sells and fits vehicle-parts to store and retrieve information about the maintenance state of its customers' vehicles and the

driving habits of their customers. Such a program might be used, for example, to decide when to send a letter to a customer reminding him that the tread on his tyres may be getting thin, or other maintenance is required, and to print such a letter. A scheduler must be able to request letters to be printed (perhaps at weekly intervals) and the system should print all the necessary letters. When maintenance work has been carried out, this should be recorded by the system and a suitable report describing the work should be generated. The customer (a `Person`) is thus an agent, and excluded from the system boundary - this is illustrated in Figure 6-5.

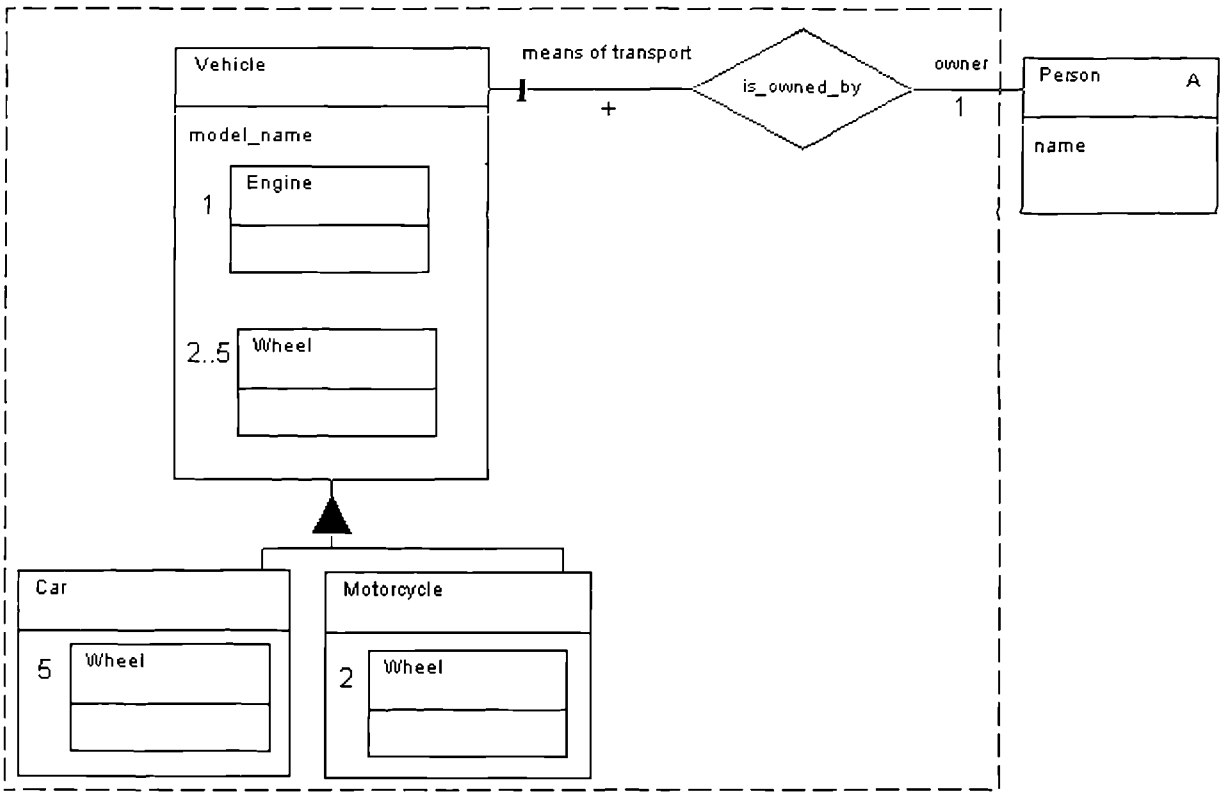


Figure 6-5: System boundary (shown dotted) and Agents (marked with A) are identified

6.2.2 Analysis: Interface Model

The interface model represents the operations which will be performed by and on the system - the transactions which must take place at the system boundary. The system operations are identified - these are the messages which the agents will send to the system - along with the agents which send them. This is illustrated in Figure 6-6.

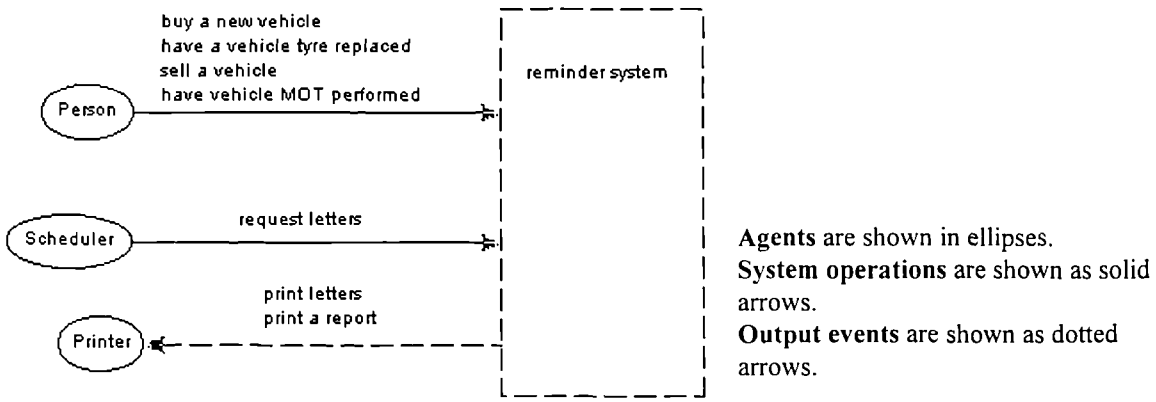


Figure 6-6: Part of Fusion interface model for the vehicle maintenance reminder system

In Fusion, for historical reasons, the message which initiates an operation is identified with the handler for it. Thus a system operation represents both the message from the agent and the actions which the system should perform in response to it. The output events are also identified, which are the messages the system should send to the agents, along with their originating agents.

The lifecycle and operation models are then produced - these are both text-based rather than graphical models. The lifecycle model represents the allowable sequences of system operations and output events. The lifecycle notation, an extended Backus-Naur Formalism (EBNF), permits sequences of system operations and output events to be arbitrarily interleaved - this permits modelling of multi-tasking systems. The lifecycle model is built up from system operation names and output event names (which are prefaced by #), or from compound expression names, using the following operators:

`A.B`	"A is followed by B"
`A|B`	"either A or B may occur"
`A ||B`	"the elements of A are arbitrarily interleaved with the elements of B"
`A*`	"A occurs zero or more times"
`A+`	"A occurs one or more times"
`[A]`	"A may occur" (i.e. it is optional)

The lifecyle model begins with the keyword "lifecycle", followed by the system name and a ":" character:

```
lifecycle <system name> :  <lifecycle expression>
```

A lifecycle expression may be given a name, and then the name may be used as an element in other life-cycle expressions (recursion is disallowed):

```
<expression name> = <lifecycle expression>
```

The operator precedence is shown below, in decreasing precedence order, and may be overridden using round brackets :

```
[ ], *, +, ., |, ||
```

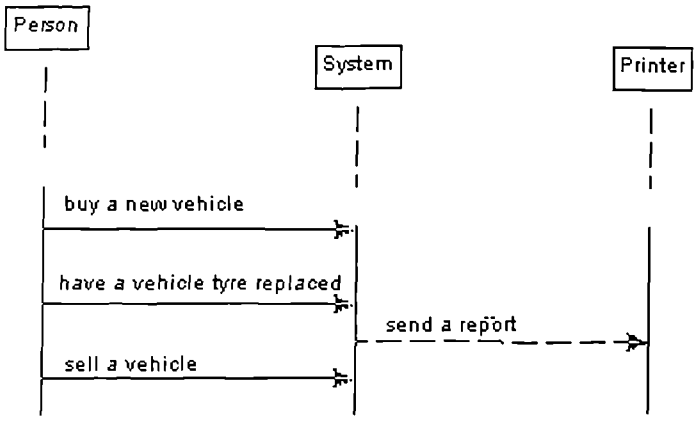
The lifecycle model for the reminder system might be as follows:

```
lifecycle reminder system : customer_lifecycle * || (request letters .
# print letters)*

customer_lifecycle = buy a new vehicle . (
 (have a vehicle tyre replaced. # print a report) |
 (have vehicle MOT performed . # print a report)
) * . [sell a vehicle]
```

The operation model is a pre- and post-condition description of each system operation based on the contractual programming model: the system undertakes to produce the post-condition given the pre-condition. There may be any number of different operation models for a single system operation, distinguished by their pre-conditions. In addition to the pre- and post-conditions, the operation model identifies which objects or values are read, changed or created by the system operation and which output events are sent. An example is shown below:

```
Operation : have a vehicle tyre replaced

Supplied : customer_name : String ,
           vehicle_number : int ,
           wheel_number : int

Returns : none

Description : " Modifies wheel to indicate that it has been fitted with a
new tyre at current time and date, and prints report. "

Changes : new report : Report,
          wheel : Wheel with " wheel_number'th wheel of vehicle_number'th
vehicle belonging to customer_name "

Sends : Printer { print a report }

Assumes : " customer_name is name of existing customer "

Result : " wheel has been modified. report has been printed. "
```

An additional graphical view is proposed in [Coleman 1994], termed a timeline diagram. In such a diagram a particular scenario, or sequence of system operation operations and output events, is described. This view is not an essential part of the system description - it is possible to deduce all legal scenarios (and there may be too many to enumerate) from the lifecycle and operation models. However, timeline diagrams do provide a useful way of checking the lifecycle and operation models. An example is given in Figure 6-7.



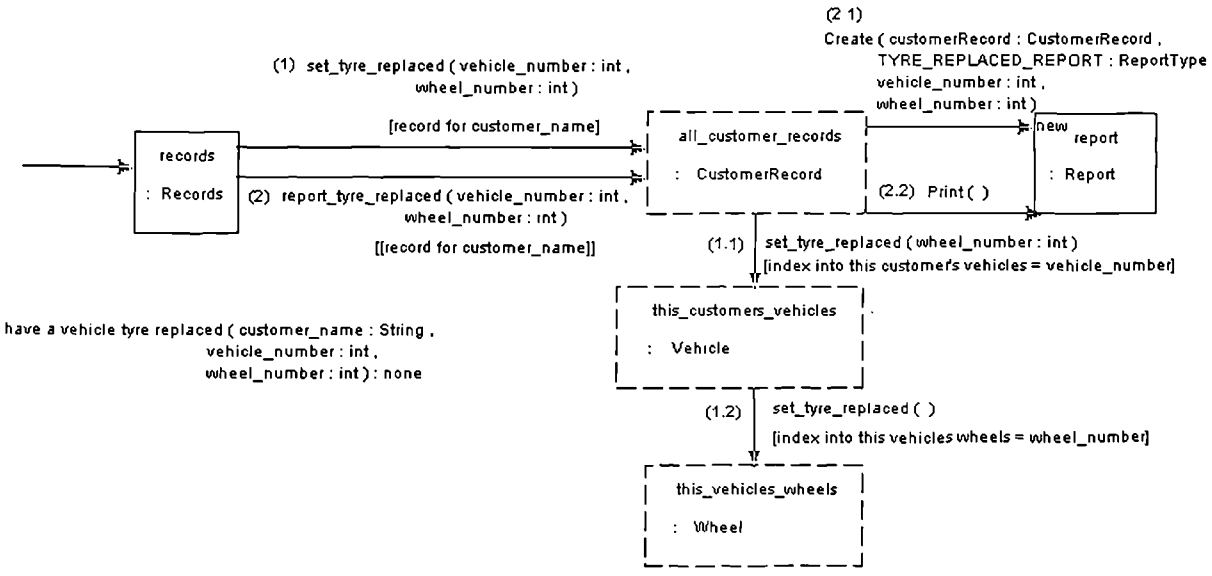
Time runs from the top of the diagram to the bottom. A solid arrow labelled <*sys op*> from Person to System, for example, indicates that Person invokes the system operation named <*sys op*> on System. Output events are shown as dotted arrows.

Figure 6-7: Example time-line scenario for the vehicle maintenance reminder system

6.2.3 Design: Object Interaction Graphs

The analysis models are now completed; the behaviour required of the system has been fully specified. The design of the system can now begin. The first step is to describe how the objects in the system should interact with each other to perform the required system operations. An object interaction graph is produced for each system operation. The controller class for the operation is chosen - this is the class which will implement the handler for the event. The pattern of method-invocation between the objects in the system which will participate in the handling of the system operation is then modelled. The sequence of methods, their signatures (defining their argument types and return type) and their source and destination objects (or collections of objects) are all represented in the object interaction graph. An operation model (or, indeed, a sub-object interaction graph) may be produced for each method on the object interaction graph. If a method is invoked on a collection of objects then a predicate must be defined consisting of a select condition and an (optional) stop condition. The method will be invoked on all members of the collection which satisfy the select condition, in an unspecified order, until the stop condition becomes true. Where necessary to resolve ambiguity, messages can be marked with a sequence number, shown in rounded brackets. All messages whose sequence numbers begin with "(N." must be completed before the message with sequence number "(N)" will complete.

An example, showing part of an object interaction graph for the system operation "`have a vehicle tyre replaced`" in the reminder system is shown in Figure 6-8.



Arrows represent method invocations (i.e. **messages**).
Solid edged rectangles represent **objects**, annotated thus <object name> : <class name>
Dotted edged rectangles represent **collections**.
Select predicates, shown in square brackets, indicate to which members of a collection a message should be sent.
Sequence numbers are shown in round brackets () thus. Repeated messages have a "*" character appended to their sequence number. If there are two alternatives for the *n*'th message, they are given sequence numbers of (*n*) and (*n'*).

Figure 6-8: Part of Fusion object interaction graph

The object `records` has been identified as the controller for this system operation. This object is distinct from the collection of customer records, since there may also be records of other types stored in the system. When the system operation is invoked, the first action (marked with sequence number (1)) is to send a `set_tyre_replaced` message to the `CustomerRecord` for the specified customer, to update the records. The second action (with sequence number (2)) is to produce a report describing what has been done by sending a message named `report_tyre_replaced` to the `CustomerRecord`. The first action involves sending a `set_tyre_replaced` message to the appropriate vehicle, and this in turn involves sending a `set_tyre_replaced` message to the appropriate wheel of that vehicle. The second action (producing the report) involves creating a new report object (using the `Create` message), and then sending it a `Print` message.

In checking the model for consistency, one should ensure that the objects identified as read or changed in the operation model appear in the object interaction graph. It is an important feature of the Fusion method that the system operations are determined (lifecycle model and system context diagram) and described (operation model) before any thought is given to which classes will handle them. This late choice of classes helps with the fundamental problem in OO design of "choosing the right classes".

6.2.4 Design: Visibility Graphs

A visibility graph is produced for each design class, identifying those server objects to which the client class must have a reference, in order to achieve the required object interactions. A visibility link, or reference, between a client class and a server object has four properties, which are all represented on the visibility graph: It may be *bound* or *unbound* - a bound server's lifetime is bound to that of the client, the server only exists whilst the client object exists. Bound servers are shown inside the rectangle representing the client, unbound servers are shown outside; in Figure 6-9, the wheels are bound to vehicle.

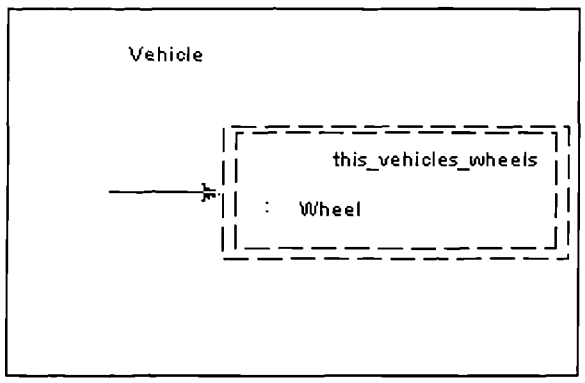


Figure 6-9: Fusion visibility graph

A link may be *exclusive* (shown as a double border) or *shared* (shown as a single border) - if it is exclusive then there are no other clients with references to the server object during the lifetime of the client; in Figure 6-9, the wheels are exclusive to vehicle. A link may be *constant* or *variable* - a constant reference signifies that the client always refers to the same server, during the lifetime of the client. The words *constant* or *variable* may be marked on the visibility graph, but in the absence of any marking links are variable by default. And, finally, a link may be *permanent* or *dynamic*. A *dynamic* reference lasts only during a single

method call - but a *permanent* reference persists between calls. Dynamic links are shown as a dotted arrow, permanent ones as a solid arrow.

6.2.5 Design: Inheritance Graphs

The inheritance graphs show the data attributes (non object valued) and methods for each design class and the inheritance hierarchy. The inheritance graph is initially built from the information contained in the other views. Opportunities for abstraction of data or methods into a parent class may present themselves during production of inheritance graphs. The notation is similar to that used in the Fusion object model (see Figure 6-10), except that the methods of the class appear in the bottom section of the rectangle, in addition to the attributes which appear in the middle section.

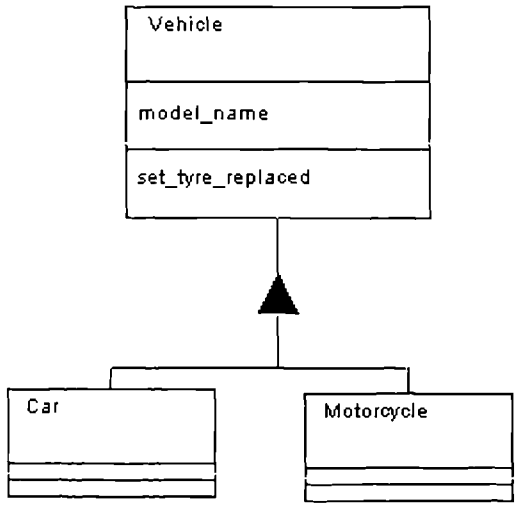


Figure 6-10: Fusion inheritance graph

6.2.6 Design: Class Descriptions

The class description schema identifies the parent class/es, attributes and methods for each design class. The attributes in the class descriptions may now include object-valued attributes (recall that this was not true of the object model or the inheritance graph). The object-valued attributes are derived from the visibility graph, the data-valued attributes, the methods and the parent class/es from the inheritance graphs. The signatures for the methods are derived from the object interaction graphs and from the operation models. The notation is self-explanatory; an example for the Vehicle classes is shown below:

```
class Vehicle isa
 attribute variable model_name  : String
 method set_tyre_replaced ( wheel_number : int )
 attribute variable this_vehicles_wheels  : exclusive bound col Wheel
endclass

class Motorcycle isa Vehicle
 attribute variable sidecar : bound Sidecar
endclass

class Car isa Vehicle
 attribute variable windscreen : Windscreen
endclass
```

6.2.7 Data Dictionary

In addition to a description of the view entities, the data dictionary should also contain descriptions of the data types (these will be language specific), any functions which are not implemented as methods, predicates (start and stop conditions for invoking methods on collections of objects) and assertions. Assertions are properties of the system which have been identified during analysis as invariant - statements about the system which must always remain true.

6.2.8 Implementation: Coding

Fusion is not directed at any one particular programming language, although implementation in C++ and Eiffel are discussed at some length in [Coleman 1994]. There are four basic tasks involved in producing the code for an implementation from a Fusion model - implementing the lifecycle, the data dictionary, the class descriptions and the method bodies.

If there is no interleaving then the lifecycle model can be implemented as a state machine, where a state represents the set of events (system operations) which may legally occur and a transition represents an event. It is a non-deterministic state machine (i.e. it can be in more than one state). If there is interleaving but no interference between the interleaved sequences (i.e. no common events) then several parallel state machines (one for each interleaved sequence) may be implemented. In the worst case, where there is interference between the interleaved sequences, then a product state machine (whose states are the cross-product of the states of the interleaved sequences) must be implemented.

The type definitions, functions and predicates stored in the data dictionary are implemented in a language-specific way. Then the Fusion class descriptions may be expressed in the implementation language - in C++ as the class definitions. The method bodies should be written using the operation models and the object interaction graphs, ensuring all assertions in the data dictionary are respected.

6.3 C++

The implementation language chosen for the risk toolkit was C++, and for this reason a brief overview of the language features is given here. C++ is a compiled language which is a super-set of C. Like C, it offers the programmer the opportunity to write code which will execute very quickly and use very little memory; the relationship between C and assembly language is closer than with other "high-level" languages such as Basic or Pascal and for this reason C is sometimes termed an "intermediate-level" language. It is possible to write C and C++ which explicitly manipulates physical addresses in memory and to optimise the assembly language instructions which the compiler will generate by looking at the C or C++ code. However, the availability of optimising compilers generally renders such effort unnecessary except when writing hardware drivers and similarly low-level code. The disadvantage of this flexibility is that the programmer is free to write "dangerous" code, and this disadvantage applies equally to C++ as to C. Where speed is of the essence however, and the risk toolkit was one such example, C++ provides a good compromise between the advantages of OO and the speed of C. It also has the advantage that C++ compilers are available for all platforms.

Unlike Smalltalk, for example, C++ does not provide object-persistence or run-time schema querying. If C++ programmers wish to store an object to disk or to display the name of the class of an object (or other meta-data about the class definitions), they must write code in order to do so.

6.3.1 Classes and Objects

In order to create an object in C++, the programmer must specify the class of the object - they may also pass arguments to the *constructor* for the class, which is a special function which is automatically invoked when a object is created. Base-typed values such as integers (`int/long/short/etc.`), floating point values (`float/double/etc.`) and characters (`char`) are also termed objects in what follows, although the C++ syntax does in some cases distinguish between base-typed values and object values. The definition of the class provides the necessary information to create the data structures in memory which implement the object. In C++ a class is basically defined by the definitions of its *data members* and its *methods* and by the identity of its *super-class* or classes (multiple inheritance is supported). A class inherits the data members and methods of its super-class es - although it will only have direct access to those which have not been "hidden" or encapsulated in the super-class definition. A *method* is a function (executable code) which may or may not return a value. *Data members* may be base-typed values or other objects, or they may be *references* or *pointers* to values or objects. They may also be arrays, but in C C++ an array is a specific kind of pointer. Confusingly, a data member may also be a *function pointer* (see Section 6.3.2 "Pointers and References", below). (Formally, these ideas are unified by the definition of an *l-value*, so called because it may appear on the left hand side of an assignment statement. In the C++ language definition [Stroustrup 1991], an object is defined as a region of storage and an l-value is defined as an expression referring to an object or function.)

The example class `MyClass`, whose definition is given below, inherits from `MySuperClass`. It has two data members `m_dX` and `m_dY`, both of which are floating point values, and a single method called `MyFunc()`, which takes no arguments (hence "void") and returns a floating point value. The method is public and thus can be invoked by objects of other classes, but the data members are private and may only be accessed by methods of `MyClass()`. `MyClass` also has a constructor - this is the function called `MyClass()`.

```
class MyClass : public MySuperClass
{
public:
        MyClass(void);
        double MyFunc(void);
private:
        double m_dX;
        double m_dY;
}
```

The class definition, above, does not include the implementation of the methods; just the definition of their names and argument types. In C++ the implementation of the methods is often stored in a separate file - the class definition, above, might be stored in a header file called MYCLASS.H and the implementation in a file called MYCLASS.CPP. The executable part of the method implementation is termed the method *body*. One of the differences between C++ and some other OO languages (e.g. Eiffel) is that most of the type-checking is performed when MYCLASS.CPP is compiled. Once all the classes used by the program have been individually compiled, the resultant object code is then linked together to yield an executable program. This

type-checking based on an "open-world" assumption, where each class is checked individually, is considered by some to be inferior to "closed world" type checking where the entire program is checked and optimised for efficiency as a whole, firstly because with C++ it is apparently possible to create programs which contain type errors and secondly because some of the burden of optimising for efficiency (where this requires knowledge about how one class is used by another) effectively has to be borne by the programmer rather than the compiler. This is explained in Section 6.3.4 "Polymorphism and Virtual Functions", below. However, one major advantage of open-world type checking is that programs can be compiled and linked rapidly - since only the class which has been modified needs to be type checked. When using large class libraries this is significant.

6.3.2 Pointers and References

The concept of a *pointer* is intrinsic to C and also to C++. A pointer is a variable which is an address in memory, often the address of another variable. The pointer may be used to read or write the value of the variable to which it points. In C++ (and C) one may declare and manipulate pointers to base-typed variables as shown below (the // characters precede comments):

```
double *pd;                 // Declare pd to be a pointer to a
                            // double precision floating point value.
pd = new double[3];         // Allocate memory for three values in a block
                            // and set pd to point to the first value.
*pd = 41.0;                 // Write 41.0 into the first value.
pd++;                       // Means pd = pd + 1. Set pd to point to
                            // the next value.
*pd = 42.0;                 // Write 42.0 into the second value.
pd++;                       // Set pd to point to the next value.
*pd = 43.0;                 // Write 43.0 into the third value.
my_function(pd - 2, 3);     // Call function, passing pointer as first
                            // argument and size of block as second.
delete [] (pd - 2);         // De-allocate the memory for the three values.
```

The example above shows how a pointer is de-referenced using the * operator. The function `my_function()` is passed the pointer, and thus can read or write data in the block to which it points. The second argument to `my_function()` informs it how much memory has been allocated; if `my_function()` ignores this information and writes a value into the fourth element of the block then this will overwrite memory which may well contain other variables, corrupting them. This is the danger of pointers.

Both C++ and C use pointers to base-typed variables, to structures (user defined aggregates of base types) and also to *functions*. When a function pointer is de-referenced using the * operator, the function is invoked. This enables *call-back functions* to be implemented. An example of the use of a call-back function is the ANSI standard C library function `qsort()` which implements the quicksort algorithm. The arguments to `qsort()` include a pointer to the data to be sorted, the number of data elements and the size of each element. But the fourth argument is a function pointer, which points to the comparison function which the `qsort()` algorithm will invoke to decide which is "largest" of two data elements. The use of call-back functions can add considerable generality to both C and C++ programs. In C++, the programmer may also declare and manipulate pointers to *objects*. Given such a pointer, the programmer may invoke the public methods of the object and edit its public data.

A *reference* in C++ is rather like a pointer, in that it is also the address of an object, but it may only ever point to one object, which must be specified when the reference is created. The programmer may not subsequently re-assign the reference to point to a different object. A reference does not need to be de-referenced using the * operator; in terms of C++ syntax, it looks exactly like an object instance:

```
double dVal;              // Declare dVal to be a floating point value
double &dRef = dVal;      // Declare dRef to be a reference to the
                          // value dVal
dVal = 41.0;              // The value of dVal is now 41.0
dRef = 43.0;              // The value of dVal is now 43.0
```

References are safer than pointers because they are guaranteed to be pointing to an object. The other main reason for using references instead of pointers is to tidy-up code which would otherwise be cluttered with * de-referencing operators. In particular, the use of references simplifies passing object-typed arguments to functions, and returning object-typed values from them. In the example below, `AFunction()` takes two references to objects of class `MyClass` as its arguments (the & operators in the function declaration indicates that they are references) and calculates and returns a value using a method belonging to `MyClass` (namely `MyFunc()`). This is faster than passing the a and b objects themselves, which would require copying all their data, but the syntax is just the same - the programmer does not have to explicitly take the addresses of a and b in the calling code and then de-reference the pointers in the body of `AFunction()`:

```
MyClass a, b;                              // Declare a and b to be
                                           /  objects of type MyClass.
double dResult = AFunction(a, b);          /  Pass references to a and b
                                           // as arguments to
                                           // AFunction.
...
double AFunction(MyClass &a, MyClass &b) // Function declaration
{
      return = a.MyFunc() * b.MyFunc() + 89.2;
}
```

6.3.3 Constructors and Destructors

The constructor is a special method which is called when the class is instantiated. The programmer may use the constructor to initialise data members, perhaps by explicitly creating other objects. The constructor may take arguments, and several different versions may be defined which take different arguments. Similarly, the destructor is automatically called when an instance of the class is destroyed. The programmer may use the destructor to delete any other objects which were explicitly created.

The copy constructor is a special constructor which takes as an argument a reference to an object of the same class, and this is invoked when the object is copied during initialisation. For example, either of the two following lines would create an object called `b`, of class `MyClass`, by using the copy constructor and passing object `a` to the copy constructor as its argument:

```
MyClass b(a);
MyClass b = a;
```

6.3.4 Polymorphism and Virtual Functions

There are (arguably) two mechanisms provided in C++ to support polymorphism. The controversial mechanism is operator and function *overloading*; the programmer may define multiple versions of method (or operator) with the same name but taking different arguments. The compiler will automatically select the version whose argument list matches the supplied arguments in the invocation. Function overloading is not truly polymorphic because the appropriate version is selected at compile-time, not dynamically at run-time. The second mechanism is function *overriding*. The programmer may re-define an existing method, belonging to a superclass, in a subclass - this is termed *overriding* the superclass method. The appropriate version of the method will be invoked at run-time, depending upon the class of the object upon which it is invoked. In C++ however, the programmer must specify which methods may be overridden using the *virtual* keyword and OO purists argue that, in a truly OO language, all methods would be virtual. The reason that in C++ the programmer must specify which methods are virtual is that virtual methods involve a time and memory overhead. If the C++ type-safe linkage were based on a "closed world" assumption where the whole program is compiled at once, then the compiler could decide which functions needed to be virtual, but since the compiler only deals with one class at a time, the programmer must perform this task.

A *pure virtual* method is a virtual method which does not have an implementation (a method body) in the class in which it defined. Classes containing pure virtual methods are termed *abstract classes* and cannot be instantiated directly; the programmer must define a sub-class which includes an implementation for the pure virtual method and create instances of this sub-class. Abstract classes provide a powerful mechanism for defining an interface between a client and a server class. The client class invokes the pure virtual methods, and thus the server class may be replaced by any sub-class without requiring changes to the client class.

6.3.5 Class Templates

A powerful feature of the C++ language is the ability to define parameterised classes where the parameter is a type (a class or a base-type). Such classes are termed class templates and are used to create families of collection classes where the type of object taking part in the collection will be known in advance (at compile time) but where the functionality of each member of the family is the same. For example, the functionality of a linked list is implemented once as a class template, `List`, and then the family of collection classes `List<int>` (for lists of integers), `List<double>` (for lists of floating point values), `List<MyClass>`, or any other parameter which the programmer chooses, can be automatically generated by the compiler. The compiler will check that floating point values are not added to a `List<int>`, and the programmer will not need to replicate the basic functionality of the linked list for each member of the family.

6.4 Summary of Decisions Concerning the Risk Toolkit

It was decided to use the Fusion OOAD methodology (supported by the FusionCASE CASE tool from SoftCASE consulting) both as an analysis, design and implementation tool for building the risk toolkit and as an object modelling language for modelling and building the user-defined classes in the risk model. The language C++ was chosen as an implementation language, for reasons of speed and portability. The Fusion class definitions for user-defined classes were to be designed graphically using the CASE tool, and the CASE tool would then automatically generate a text file containing class definitions in Fusion syntax. A C++ code

generator would be built which took the Fusion class definitions as input and created a C++ class for each user-defined class. In the final implementation, the method bodies for these C++ classes were built by hand (see Chapter 10 for more details). The user defined classes would then be compiled and linked with the remainder of the risk toolkit code, to produce a version of the risk tool tailored to the user's design domain. This risk tool could then be used to build the risk model by creating instances of the user-defined classes, editing their data and creating links (relationships) between them. The risk tool could also be used to browse the risk model and to evaluate it and display the results. The risk model would automatically support uncertainty representation (i.e. without the user having defining any special Fusion classes concerned with uncertainty representation).

Despite the apparent desirability of using an OO DBMS to provide object persistence and run-time schema-querying, it was decided for reasons of cost and technology immaturity to provide support for these functions using custom code.

6.5 References

[Armstrong 1994] Armstrong, J. M. and Mitchell, R. J., "Uses and abuses of inheritance", *Software Engineering Journal*, vol. 9, no. 1, pp. 19-26, Jan. 1994.

[Booch 1991] Booch, G., "*Object-Oriented Design With Applications*", Benjamin Cummings, Menlo Park, California, 1991.

[Coad 1991] Coad, P. and Yourdon, E., "*Object-Oriented Design*", Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[Coleman 1994] Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F. and Jeremes, P., "*Object-Oriented Development - the Fusion method*", Prentice Hall International Editions: Englewood Cliffs, New Jersey, 1994.

[Dock 1992] Dock, P., "OOD: Research Or Ready?", *Hotline On Object-Oriented Technology*, Vol. 3, No. 9, July 1992.

[Eastman 1994] Eastman, C. M. and Fereshetian, N., "Information models for use in product design: a comparison", *Computer-aided Design.*, vol. 26, no. 7, pp. 551-572, July. 1994.

[Malan 1996] Malan, R., Letsinger, R. and Coleman, D., "*Object-Oriented Development at Work : Fusion in the Real World*", Prentice Hall PTR, 1996.

[Meyer 1989] Meyer, B., "*Object-Oriented Software Construction*", Prentice-Hall, 1989.

[Meyer 1992] Meyer, B., "*Eiffel: The Language*", Prentice-Hall, 1992.

[Murdoch 1994] Murdoch, T. and Ball, N., "The development of an EDC product data model", *Internal Report, CUED/C-EDC/TR21*, Cambridge Engineering Design Centre, pp. 1-22, December. 1994.

[Rumbaugh 1991] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W., "*Object-Oriented Modeling and Design*", Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[Schenk 1994] Schenk, D. A. and Wilson, P. R., "*Information Modelling: The EXPRESS Way*", New York : Oxford University Press, 1994.

[Stroustrup 1991] Stroustrup, B., "*The C++ Programming Language, Second Edition*", Reading Massachusetts, USA : Addison Wesley, 1991.

CHAPTER 7

Case Studies of Present Design Practice

In this chapter, case studies are introduced from two of the industrial collaborators - Cegelec Projects and Rover Group. Each case study describes a particular early design project - a steel roughing mill and a dynamic positioning system for Cegelec Projects and the interior trim for a new vehicle development at Rover Group. The purpose of the case studies was to study the types of design domains where the risk methodology and tool should be applicable and to provide examples to test the generality of the modelling methodology - illustrating, for example, how cross-coupling between different branches of a functional decomposition could be modelled and how different companies' ideas of how to model management and administrative costs should be handled. In Chapter 11, these case studies are modelled and evaluated using the risk tool and the results are presented.

7.1 Tendering for Large Scale Electrical Installations at Cegelec Projects

Two simple case studies are presented in this section, both of which are based on "typical" projects - however, when the results are presented in Chapter 11, no significance should be attached to the numerical values used, which in no way reflect the actual costs for such projects. The case studies concern cost-risk estimation in the early design phase for large-scale control installations - where the "early design phase" consists of preparation of a tender for a bid. Both case-studies concern the cost to supply the project without any further development, other than the provision of software, (termed Costasis in Chapter 4), since this will be the basis for the price quoted in the tender document - separate costs for additional development work under consideration (as described in Chapter 4) were not calculated in these case studies.

7.1.1 Steel Roughing Mill

The first case-study involves building up a cost-risk estimate for a roughing mill (a process area in a steel works), for inclusion in a tender document. This case study introduces the key attributes and the decomposition hierarchy used for control installation projects at Cegelec Projects, however it is a highly simplified example which was developed largely in order to test the results obtained from the risk toolkit against those generated using a commercial Monte Carlo simulation package (see Chapter 11).

During early design, the control installation project is hierarchically broken-down into its constituent functions (functional decomposition). At the highest level, the project is decomposed into the process areas which are required. This case study concerns one such process area - the steel roughing mill. At the next level of functional breakdown, the process area is decomposed into the types of function which are required - for the roughing mill these were Control (which includes both software and hardware), Cabling and Management. Each function type is then further decomposed into functions and sub-functions. During

preparation of the tender, there may be several alternatives under consideration for any of the functional nodes. The functional decomposition for the roughing mill is shown in Figure 7-1.

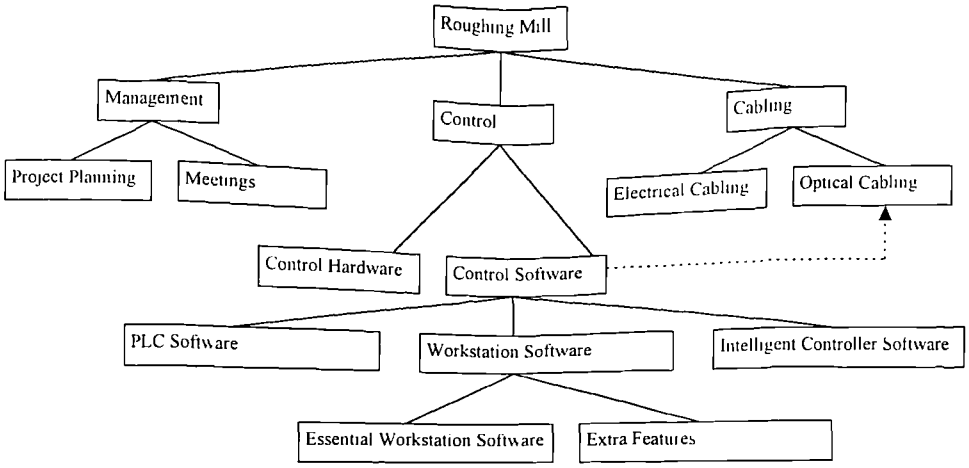


Figure 7-1: Functional decomposition for roughing mill tender

There are three different kinds of function (nodes in the breakdown) which may be required, each incurring a different type of cost, all of which are uncertain. The uncertainty in cost is represented using an integer "function development status", as described in Chapter 4, which is interpreted as providing the upper and lower limits on an interval as a percentage of an estimated cost value. Hardware (for example Cabling, or hardware required to provide the Control function) has an associated hardware cost which is estimated directly. Software functions have an estimated number of weeks of effort required to provide the function, and this is then combined with an estimate of the loaded rate for software programmers to yield the software design cost. Similarly, Management functions have an associated number of weeks which is combined with a loaded rate to yield the Management cost. It follows that a fourth type of function exists, for example the Control function, which may contain a mixture of Management, Software and Hardware sub-functions and thus has all three types of cost associated with it. Integration costs are also important. The number of weeks required to integrate several software modules together may be significant - and is evaluated - but this is included in the total weeks of effort required for the combined software package, and hence is counted as part of the software cost. However, the cost of integrating software with hardware is regarded as a fourth separate category of cost. Each of the four categories of cost may be evaluated separately at any level in the functional breakdown, or their total may be considered. Similarly, the total number of weeks of software effort or of management effort may be evaluated at any level in the functional breakdown.

Figure 7-2 illustrates the objects and attributes discussed so far, using the Fusion object-oriented graphical notation for an object model (see Chapter 6). The class PhysicalObject represents hardware functions, CegSW represents software functions, CegMan represents management functions and CegProduct represents functions which are a composite of hardware and/or software and/or management. There is some duplication of attributes - for example man_cost and man_weeks occur both in CegProduct and in CegMan. This could have been avoided by inheriting CegProduct from the other three classes but this would have required the use of multiple inheritance which is not supported by the risk model for the reasons presented in Chapter 6.

As mentioned above, the cost of integrating software modules is significant, and a distinction is made

between "heterogeneous" and "homogeneous" integration costs. The homeogeneous integration cost is defined as the extra cost incurred per component when integrating more than one component of the same type. The heterogeneous integration cost is the cost of integrating N of these components with other types of

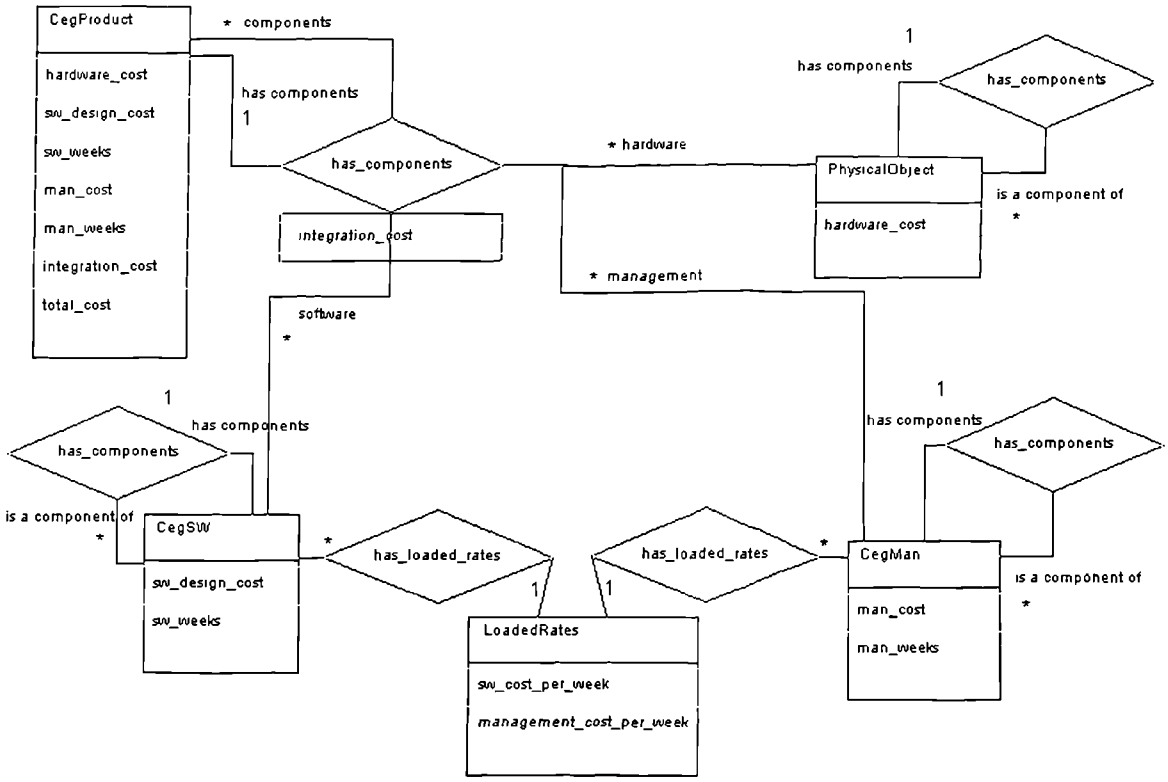


Figure 7-2: Part of Fusion Object Model for functional breakdown of a control installation

component. Also, when several components of the same type are to be integrated, there may well be "economies of scale" which can be achieved in the costs of the components - the first module will typically involve far greater development time, and thus have a greater cost, than subsequent similar modules with which it will be combined.

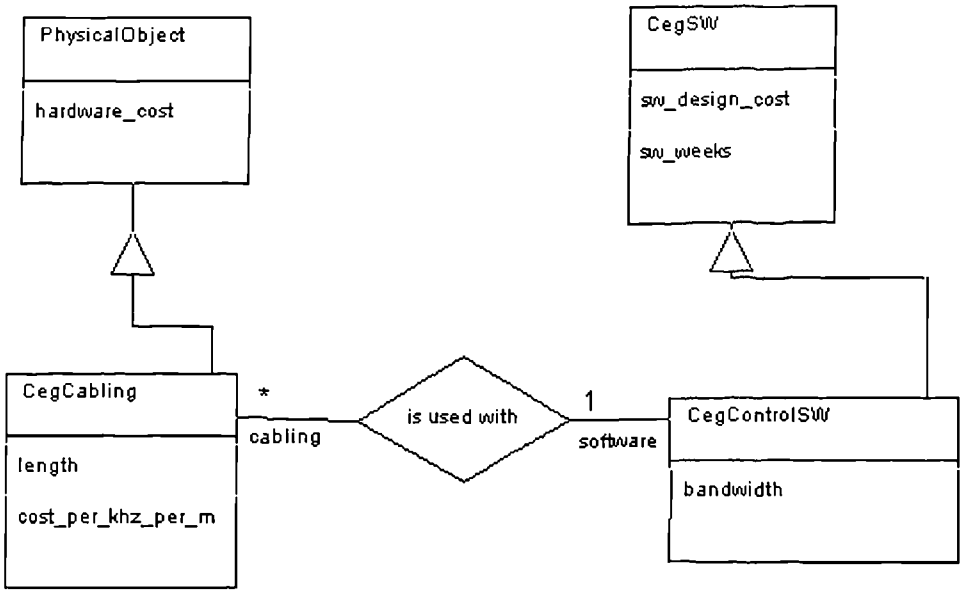


Figure 7-3: Fusion object model showing example of cross-branch dependency for roughing mill

In a functional breakdown of a tender, there will typically be some dependencies between different branches

of the hierarchy. In the roughing mill, for example, the cost of the optical cabling per unit length may increase as the bandwidth required increases, and the required data bandwidth may be determined by the control software, thus creating a dependency between an attribute of Control Software (bandwidth) and an attribute of Optical Cabling (cost), shown as a dotted line in Figure 7-1. The objects and attributes for this simplified example of a cross-branch dependency are shown in Fusion notation in Figure 7-3.

7.1.2 Dynamic Positioning System

The second case-study involves evaluation of cost-risk for a Dynamic Positioning (DP) system (a shipboard control system), again for inclusion in a tender document. The functional decomposition is shown in Figure 7-4, overleaf, and contains approximately twice as many functional nodes as the *roughing mill case study* and is thus more realistic in terms of scale. The nature of the the decomposition and types of functional node are the same as for the roughing mill. There are no cross-branch dependencies in the DP case study.

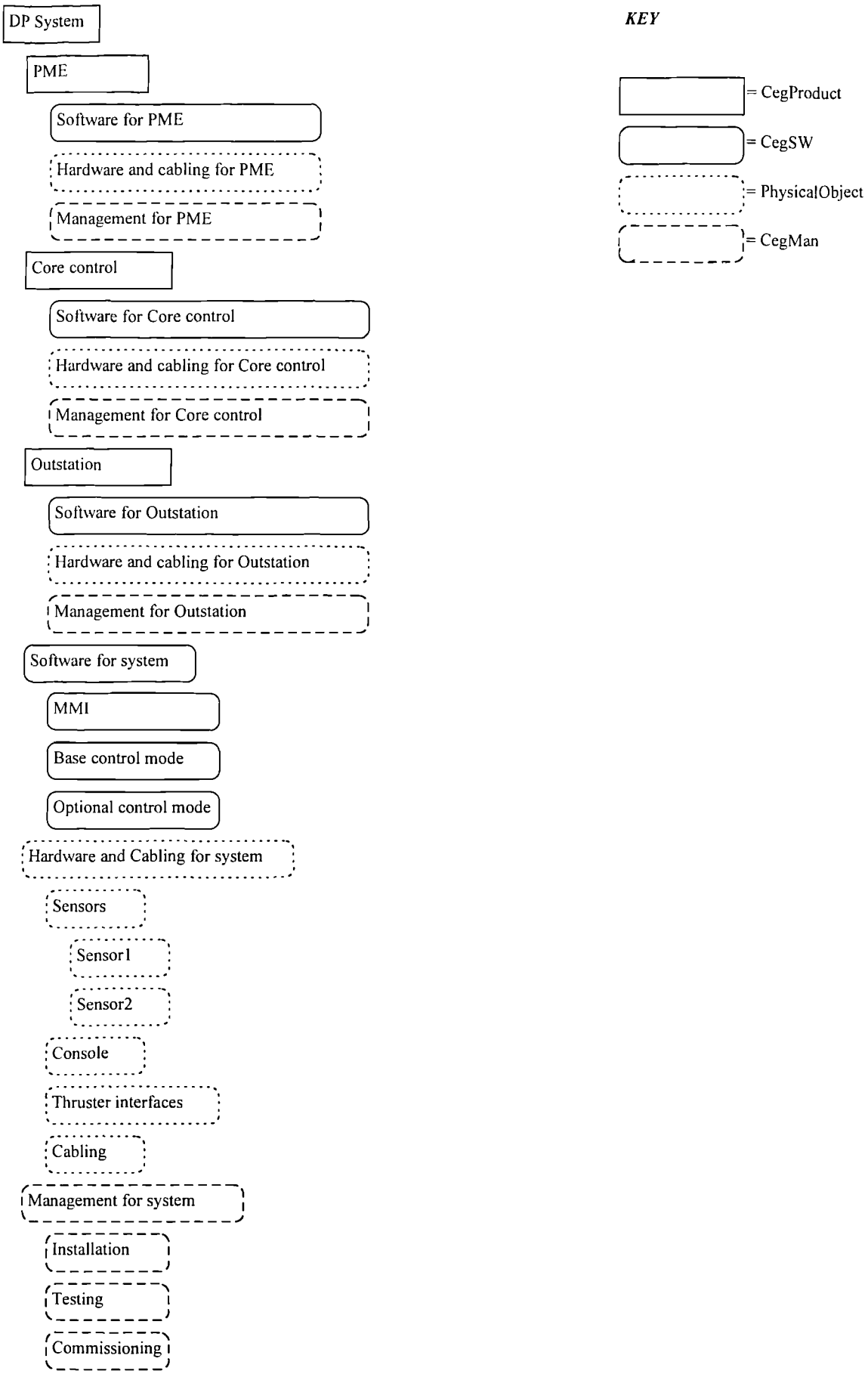


Figure 7-4:Functional decomposition for dynamic positioning system tender

7.2 Cost-Risk in Early Design of Interior Trim Area at Rover Group

This case study concerns a recent design project, during which the risk tool was evaluated using "live" data. The numerical values and some details have been alterered to protect commercial confidentiality, and the results are presented in Chapter 11. In this section, the background to the study is presented - we describe the design environment and the nature of the design decisions faced. The case study concerns the early design phase of a portion of one area within a new vehicle program. The vehicle is decomposed into five areas during the design process; Chassis & Suspension, Body, Trim, Electrical and Power Train; and the designers are divided into teams according to vehicle area. The Trim area is typically further sub-divided into Interior and Exterior and this case study concerns the Interior Trim area which includes the floor-coverings, the interior roof linings, the seats and seat covers and seat belts, the heating system and the facia assembly.

The early design phase in a new vehicle programme begins with the concept definition and ends at a time known as "D-zero", when a feasibility decision is taken. If the design is judged to be feasible at D-zero, in terms of criteria which include manufacturing costs, capital investment costs for the assembly-line tools and performance parameters, then part numbers are allocated and the detailed design proceeds. If not, the vehicle programme may be halted and never taken into manufacturing. By D-zero, the suppliers are usually bound to an agreed cost target for each bought-in component. Typically lasting for up to two years and involving over 100 designers by the time D-zero is reached, the early design phase accounts for a substantial proportion of the overall design costs.

The three key cost attributes which are developed during early design are the piece cost, the logistics cost and the tool cost. These are the multiple criteria by which alternatives are typically compared and the overall feasibility of the vehicle program is largely dependent on the overall values of these three cost attributes. Of course, all three cost attributes are uncertain during the early design phase. The piece cost is the cost incurred for each instance of an assembly, sub-assembly or component which is manufactured - this includes the material costs, the manufacturing processing costs and any sub-component costs. The logistics cost is also incurred for each instance of the part which is manufactured; this is the cost associated with packaging and transporting the part. The tool cost is the total cost associated with preparing to manufacture the part, for example the provision of machine tools and assembly stations, and thus is only incurred once regardless of how many of the parts are manufactured. Estimates for the cost attribute values are all based on an assumed production volume for the part - thus production volume estimates are also developed and recorded during the early design phase. The production volume will generally have a strong impact on many of the costs, with high production volumes leading to high tool costs but low piece costs.

The design engineer (or specialist vehicle estimator) has accumulated, through experience on many projects, much knowledge which is used directly or indirectly to provide estimates of the costs, weight and other attributes of a part. Two types of knowledge which are particularly applicable (to cost for example) are historical information concerning the costs of similar parts which have been manufactured previously and heuristic methods ("rules of thumb") which can be used to provide approximate cost values from a few simple, high-level attributes of the part. For example, an experienced vehicle cost estimator may be able to estimate the costs for a glove box moulding from its crude dimensions and the kind of trim (leather or soft

feel plastic or hard feel plastic) being used.

During early design, budgets (or targets) for piece, tool, and logistics costs are allocated to each area and to each reasonably sized assembly within an area. These target values do not remain fixed throughout the early design phase; they are periodically re-allocated on the basis of the current state of the overall design. If for example, one assembly is substantially within its piece cost budget, then part of its budget may be re-allocated to another assembly which is proving more expensive to manufacture than anticipated.

A fourth attribute which is highly significant during early design is weight; this too is aggregated over the entire vehicle and budgets (or target weights) are allocated to each vehicle area and within each vehicle area. In this case, the reason for monitoring the overall weight is not direct impact on profitability (as for the cost attributes), but the impact of the weight on the choice of suspension system (and hence its cost), on the engine power required and on performance parameters such as fuel consumption, acceleration and maximum speed. Weight is not a relevent attribute for all parts, however, particularly in the Trim area; only mechanical parts have weight. The vehicle owners handbook or the roof trim provide examples of parts which do not require a weight attribute.

A distinction is made between components which will be designed in-house and those where the responsibility for calculating or estimating the costs has been delegated entirely to the supplier. For such bought-in components, the quotes given by the supplier are recorded. An example would be the sound system - the automotive designer would be unlikely to attempt to model the manufacturing process etc. for such an item.

Figure 7-5 illustrates the objects and attributes discussed so far, as a Fusion object model. Components, Assemblies and BoughtInComponents all inherit the three cost attributes and a production volume from a base class called PhysicalObject. Each assembly may have zero or more physical objects as its components - thus an assembly may be composed of any combination of assemblies, components and bought in components

Sometimes the cost attributes are initially directly estimated by a specialist vehicle cost estimator from sketches prepared by the designer. More often, however, the material costs and manufacturing process costs are estimated first and the cost attributes for the part are then calculated from these. The object model shown in Figure 7-6 illustrates the attributes and relationships for materials and manufacturing processes.

Each component is generated by zero or more manufacturing processes. Each manufacturing process has a material cost and a processing cost, which are incurred each time an instance of the part is manufactured and a tool cost which is only incurred once. An assembly is constructed by zero or more assembly processes - an assembly process is a specialisation of a manufacturing process, and consumes zero or more materials. Figure 7-6 illustrates the distinction made between a component and an assembly - a component cannot be further decomposed and is made from a single material, whereas an assembly may have many components - each one made from a different material. A part with more than one material is regarded as an assembly - unless the additional material can be regarded as a part of the assembly process, for example when painting a

door panel. The exception to this rule is bought-in components - any item which is purchased from a supplier as a single entity is considered to be a bought-in component, regardless of its physical make-up.

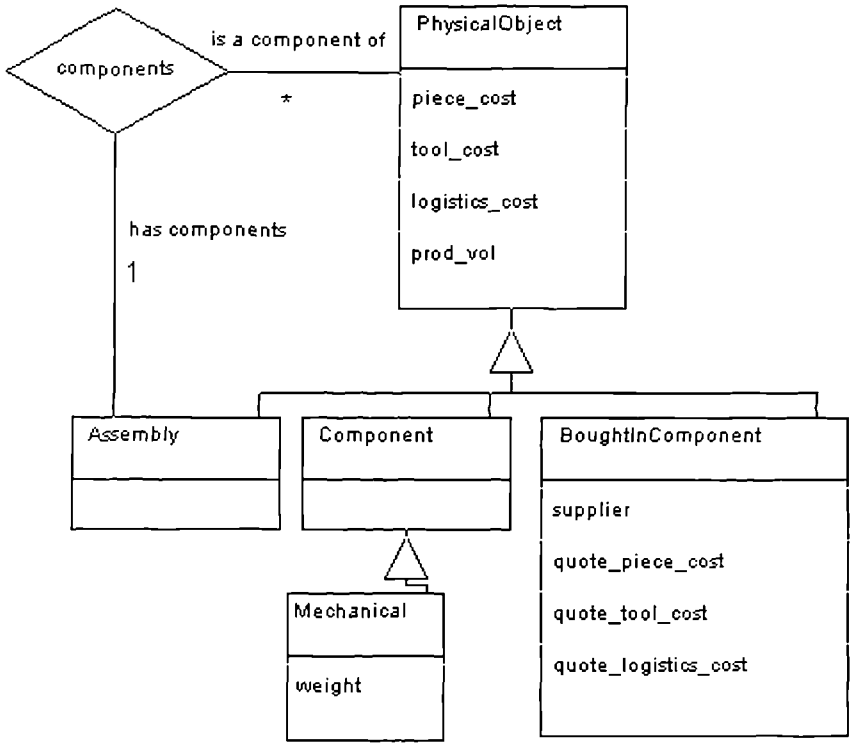


Figure 7-5: Part of Fusion Object Model for Components and Assemblies

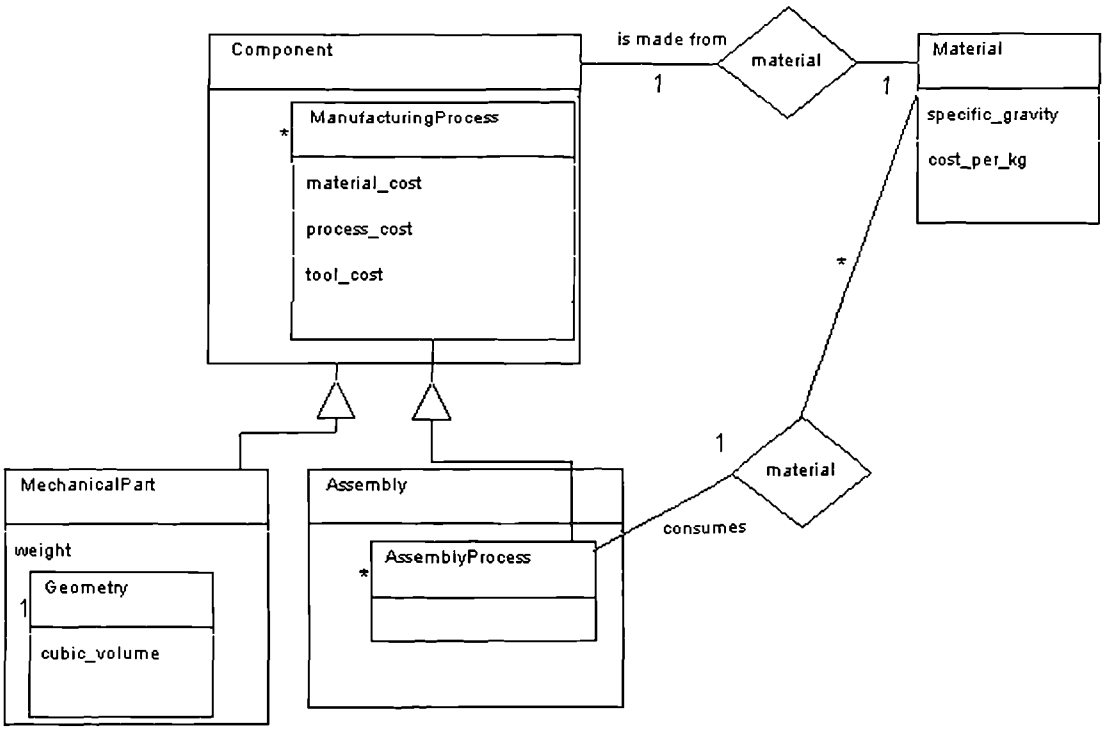


Figure 7-6: Object Model for Materials and Manufacturing/Assembly Processes

Thus, during the early design phase of the interior trim area, estimated values for cost and weight attributes of components and assemblies are gradually built up by the design team. A "Bill-of-Materials"-like structure for the interior trim area is incrementally developed and populated with attribute values as the design

proceeds. Cost estimates may be obtained from the engineer responsible for the part, from a specialist vehicle cost estimator, from an actual or potential supplier's quote or from the cost of a similar previously manufactured part. Several alternative suppliers will often be considered for a particular part, and quotes obtained from all of them before a decision is made. Several alternative manufacturing processes are also often explored, particularly since the component production volumes are usually highly uncertain during this early design phase, and the choice of manufacturing process for a part is largely dependent on its anticipated production volume.

As well as recording estimated costs against parts, the engineers in the design team also record a list of "risks" (and "opportunities"). These are uncertain costs which may possibly be incurred (or saved) in the future - usually by changing the part in some way. Risks and opportunities may be recorded for piece, tool or logistics costs.

It is well known that vehicle manufacturers offer their customers a bewildering range of products. The BMW 525, for example, has 8,589,934,592 possible configurations - it could be manufactured for 4000 years without any two cars rolling off the production line having the same configuration. This means that very many components and assemblies are designed and manufactured in several different variant forms - and even during the early design process, the variant components and assemblies which will be required, and their impact on costs, are considered. The existence of several variants for a particular component reduces the production volume for each component variant and thus, generally, increases the piece cost. The provision of extra tools to build the variants will clearly also increase the tooling cost. Thus the "configuration design" - the structure of configurations offered and how they are built from variant parts - has a strong impact on both cost and, since sales volumes are highly uncertain, risk. During the early design process, the costs for a representative sample of the product offerings within the vehicle program are usually evaluated at the highest level - perhaps ten or twenty different vehicle models may be considered.

Figure 7-7 illustrates, in rather a formal way, the classes and relationships required to represent the impact of variants on a Bill-of-Materials view of the vehicle program. The entire program is shown as the class ProductModel, which contains the ComponentModel (a Bill-of-Materials breakdown of assemblies and components) and the FeatureModel, which contains a description of each product offering under consideration. There is a many-to-many relationship between products and features - each product provides many features and a feature may well appear in more than one product. Features might for example be "sporty trim" or "fuel injection engine". Some of the component and assemblies in the Bill-of-Materials are members of sets of variants. The definition of such a set is that its members will all be manufactured but a maximum of one member of the set will be included in any one particular product instance. Also, all the members of a set of variants fit into the Bill-of-Materials hierarchy in the same place and sets of variants are always disjoint. The presence or absence of a feature determines which of a set of variants is "selected" for inclusion in a particular product offering. The engineers in the design team generally develop design information for a whole set of variants even in the early stages of the design process.

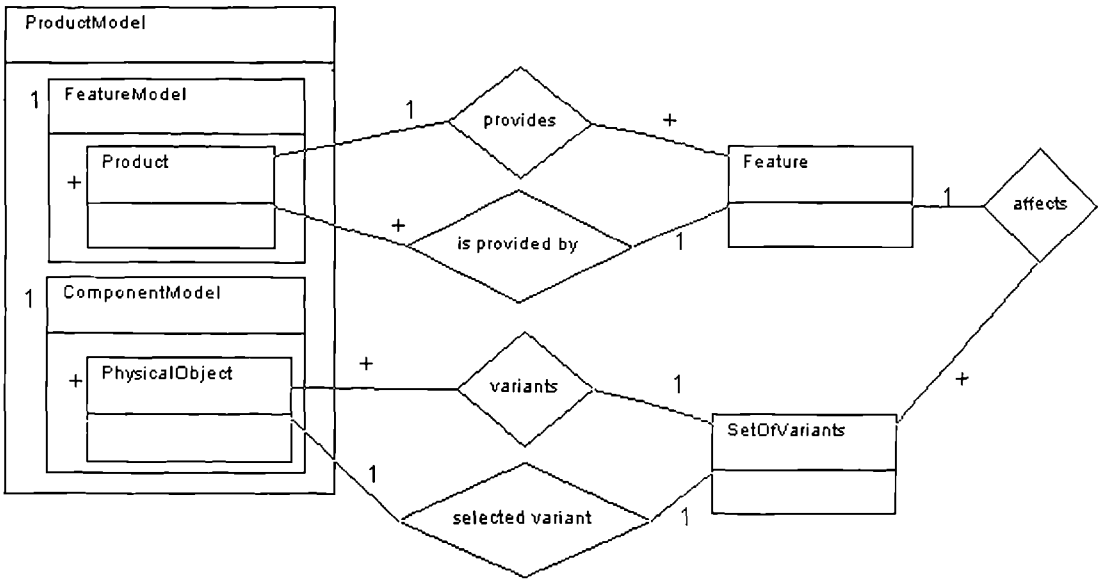


Figure 7-7: Object Model for Variant Assemblies and Components

An example of the type of document which was used by the design engineers to monitor component cost, prior to and during the period of the case study, is given in Figure 7-8.

COST PACK
ATTRIBUTES WHICH AFFECT COST

ROVER GROUP

MODEL ❶ AQUARIUS	PAGE ❷ 1	OF 1	ISSUE A	WRITTEN BY ❸ BJ POICE	
ILLUSTRATION ❹				PHONE: ❺ 053 1234	DATE: ❻ 5-9-91
				MATERIAL ❼ POLYPROP - HIMOUNT SP10	
				PROCESS MANUFACTURE ❽ (IN HOUSE/SUPPLIER) INJECTION MOULDING / GRAINED FINISH	
				PROCESS ASSEMBLY ❾ (IN HOUSE/SUPPLIER) PARTIAL ASSY BY SUPPLIER	
				SUPPLIER ❿ T.B.A.	
				PIECE COST ⓫ (VCE /ENGINEER) VCE ESTIMATE £12-08	
				WEIGHT ⓬ 4.5Kg ESTIMATED	
				TOOL COST ⓭ (VARIOUS PHASES) £500,000 (INC. GRAINING)	
PART DESCRIPTION ⓮ FACIA MAIN MOULDING	PART NUMBERS ABC 123456 XXX			VOLUME ⓯ (PEAK YEAR/LIFETIME) 60,000 PEAK YEAR	
				ISSUE CADCAM DETAILS ⓰ 3D CAD MODEL	

NOTES ⓱
NON HANDED COMPONENT
TWO TRIM COLOURS

Figure 7-8: Example of Cost Pack form

The cost pack is the first document to be developed and is completed by the design engineer - although the cost estimates may be provided by a specialist vehicle cost estimator (VCE). Estimated piece and tool costs are recorded, along with the material, the manufacturing and assembly processes, estimated weight, the

proposed supplier if appropriate and the production volume on which the estimates were based. Several updated versions of the cost pack will typically be generated as the design proceeds.

7.3 Summary

An analysis of the design domains for the case studies has yielded simple Fusion object models illustrating the main object classes and attributes involved in each study. The Cegelec case studies involve cost-risk estimation during preparation of a tender for a bid. The first, simplified, Cegelec case study (of a steel roughing mill), and the more realistically-sized second study (of a dynamic positioning system) are both modelled using the same set of classes - namely, PhysicalObject (hardware), CegSW (software), CegMan (management tasks) and CegProduct (a combination of the first three). The tender is functionally decomposed into a hierarchy whose nodes belong to one of these four classes. Four types of cost may be associated with a node in the hierarchy - software design cost, hardware cost, management cost and integration cost (cost of integrating software with hardware). Software and management costs are evaluated from estimates of the number of weeks of effort required. All four types of cost may be evaluated at any level in the hierarchy, and uncertainty is modelled using the "product development status" of Chapter 4. The cost of integrating software modules with one another is also modelled using heterogeneous and homogeneous integration costs and a factor to take account of economies of scale when integrating several modules of the same type. An example was given of a "cross-branch" dependency between the cost of optical cabling and the bandwidth required by the control software.

The Rover case study involves the early design phase of the interior trim area in a new vehicle program - the whole program typically involving over 100 designers over a period of two years, the early design phase terminates at "D-zero", when a decision is taken over whether or not the programme will be taken into manufacturing. The case study is modelled using several classes which are types of PhysicalObject, arranged into a Bill-of-Materials-like hierarchy - in particular Assembly, Component, BoughtInComponent and MechanicalPart. Important, and uncertain, attributes for all of these classes include piece, tool and logistics costs and also production volume. Materials and manufacturing assembly processes are also modelled for each node in the Bill-of-Materials breakdown. Even at this early stage in the design process, work has begun on the configuration design - the different configurations which will be offered to the customer and how they are built from variant parts - and this is intrinsic to a cost-risk assessment. Sales volumes are notoriously difficult to predict and their impact on production volumes and hence costs depends upon the configuration design. Classes ProductModel, FeatureModel and ComponentModel are introduced to model the configuration design.

This chapter has provided examples of the types of design domains where the risk modelling methodology should be applicable - the methodology is presented in the next chapter. The suitability of the risk modelling methodology and tool for representing the designs in the case studies is explored in Chapter 11, where the results of implementing the models are presented.

CHAPTER 8

The Design Modelling and Risk Assessment Methodology

This chapter presents the modelling methodology which has been developed. The methodology allows teams of designers to store abstract, uncertain or incomplete design information into a product design repository (or risk model), as the design proceeds. The main aims are threefold: To obtain early estimates of the costs, performance parameters, time-scales or other design attributes, which include the probability of achieving them; To be able to identify the major sources of uncertainty in the design and thus determine where investment is most likely to result in reducing the uncertainty; and to be able to monitor overall risk levels as the design develops. In order to achieve these aims, it must be possible to build and maintain the risk model quickly and easily with the minimum of interference with, or distraction from, the design process itself; and to this end an object-oriented approach is adopted. The use of object classes supports the re-use of modelling and cost estimation (or other attribute estimation) experience. The designers are supported in quickly and simply building detailed risk models by the provision of object classes which encapsulate knowledge about the design domain. The risk model can be incrementally refined in several ways to reflect the development of the design.

8.1 Overview

The designers build the risk model by creating, deleting and editing objects which are termed Sim objects (so called because it is possible to evaluate their attributes by Monte Carlo simulation - the evaluation method is not discussed further here as it is not relevant to the methodology, but see Chapter 9). Sim objects may represent any part of the design - for example the components and assemblies in an automotive design, or the tasks to be performed during the process of designing software, etc. In order to create a Sim object, the designer need only specify its class. Pre-defined classes encapsulate knowledge and experience - for example heuristic methods, historical information and default values - relevant to the class of Sim object. Thus, by developing a library of classes, a company-wide and company-specific repository can be built up containing knowledge about cost-risk (and other kinds of risk) for the types of entity which the company designs.

Sim objects differ from traditional product design objects in several significant ways, and this enables them to represent the types of uncertainty which are present during early design. This is discussed in more detail below. In summary however:

- Numerical attribute values may be uncertain and can be specified by the designer using a probability distribution with its parameters. Attribute values may also be completely unknown.

- Links between Sim objects may be uncertain - this represents the situation where several alternatives are under consideration only one of which will eventually be selected by the designer. Link values may also be completely unknown.
- A numerical attribute value may have several different *derivation routes* (ways of evaluating the attribute) specified for it, and they will be used in order of preference, as the necessary information becomes available, during the design process.
- *Heuristic*, or approximate, *methods* can be defined. A *heuristic method* returns a probability distribution and its parameters given point-valued inputs (unlike a deterministic method which returns a point value given point values as inputs).
- Where several *variants* of a component or an assembly exist, these can be included in the model.

Relationships between parts of the design (e.g. components and assemblies) are themselves represented as objects. There are three classes of relationship:

- ICO (is a component of). This is the basic parent-child relationship between an assembly and its sub-assemblies, a task and its sub-tasks etc.
- IAO (is an alternative of). This type of relationship is used to represent uncertain links between Sim objects, where there are design alternatives under consideration.
- IVO (is a variant of). This is the relationship between the members of a set of variant parts, where all of the members will be manufactured but only one member will be included in a particular configuration of the designed artefact.

New classes of relationship may be derived from these three base classes, for relating specific classes of Sim objects.

8.2 Classes

A set of pre-defined classes (termed the *base classes*) are provided and these can be simply used as supplied. However it is also possible to create new, company-specific classes, which are specialisations of the base classes and encapsulate the company's knowledge about the design domain. The base classes are arranged in a single-inheritance hierarchy, most of which is shown in Figure 8-1, and most of the base classes are derived from the root class `Sim`. A full description of all the base classes is given in Appendix D. A `Sim` object has *attributes*, *links* and *methods* given by the class definition.

An *attribute* is base-type valued (a floating point value, an integer, a Boolean value or a string) and a *link* is object-valued - it is a pointer to another object. A *method* is an executable piece of code which returns a numeric value. A class inherits attributes, links and methods from its parent in the inheritance hierarchy. For example, `SimWithID` provides attributes called `name`, `description` and `identifier` to all classes which inherit from it.

The class definition also determines the cardinality of the attribute or link - this may be a fixed value (for example the class `PhysicalObject` has an attribute called `piece_cost` with a cardinality of 1) or may be variable (the class `AssemblyProcess` has a link called `material` which has variable cardinality).

As well as attributes and links, the class definition also determines the methods provided to a `Sim` object by its class. Methods are used to calculate values for attributes. There are two kind of method - *deterministic* and *heuristic*. Most methods are deterministic. A *deterministic method* is an executable piece of code which collects its inputs using the attributes and links of the object to which it belongs, performs a calculation using these inputs and returns a value. The value returned may be an INTEGER, BOOL or FLOAT. A *heuristic method* takes point values as inputs and produces an uncertain value as output - thus the method itself is a source of uncertainty in the model. Heuristic methods are described in more detail in Section 8.7 below.

8.3 Sim Objects

There are three main kinds of object in the risk model (see Figure 8-1) - `Sim` objects, `UncertainValue` objects and `IAO` objects. `UncertainValue` objects represent probability distributions and their parameters and are all of class derived from `UncertainValue`. `IAO` objects represent is-alternative-of relationships between alternative `Sim` objects only one of which will eventually be chosen, and are of a class derived from `IAO`. Figure 8-1 shows two `IAO` classes - `RandomIAO` and `ByVolIAO` (see Section 8.6 below). The class `Editable` provides object persistence and run-time schema querying.

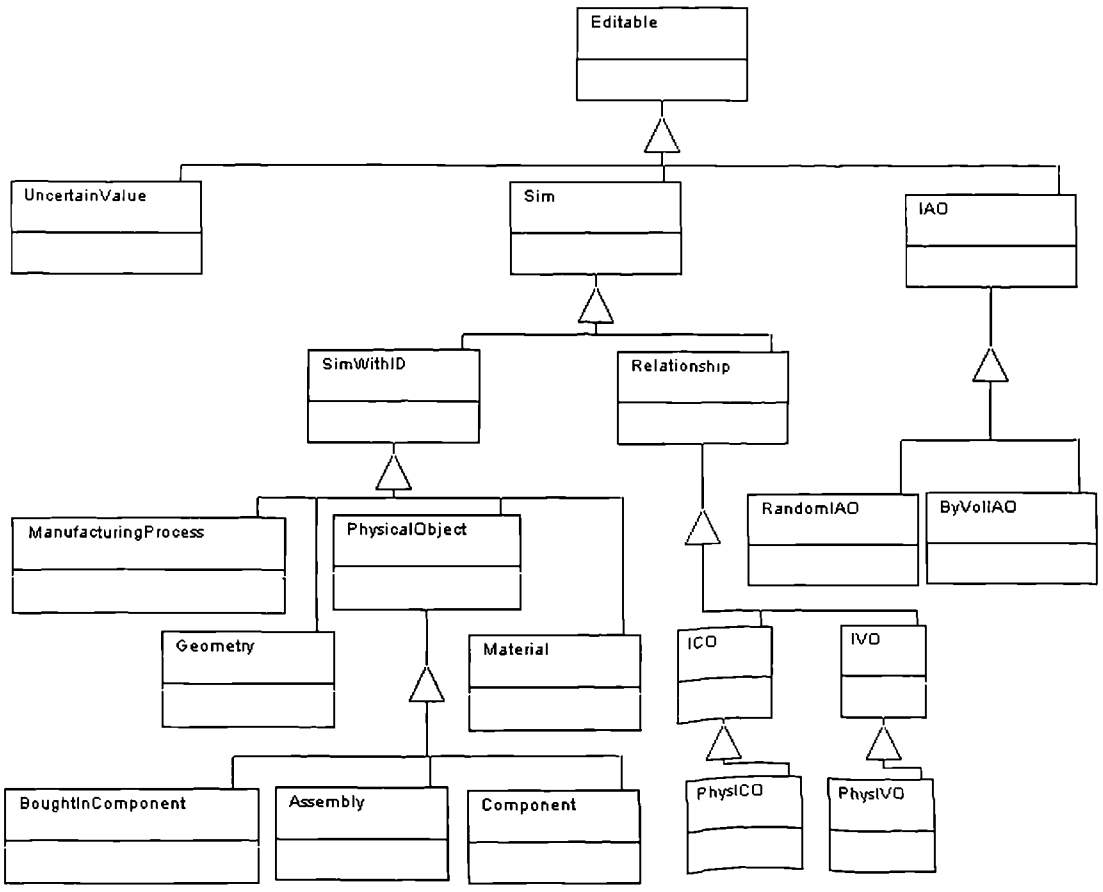


Figure 8-1: Part of inheritance hierarchy for base classes

A `Sim` object is an instance of a class derived from `Sim` with *derivation routes* for the links and attributes. The way in which these routes may be defined supports uncertainty modelling.

A *derivation route* for an attribute value may be:

- UNKNOWN
- a point value
- an uncertain value (a pointer to an `UncertainValue` object)
- a method on a `Sim` object which is reachable from this `Sim` object by traversing links
- another attribute on a `Sim` object which is reachable from this `Sim` object by traversing links

It can be seen that these are referred to as derivation routes rather than values since in general they define a way of obtaining a value, rather than the value itself. Derivation routes are similar in purpose to the "derivation" entities in the stage-based model of the variant design process developed in [Lehane 1990]. Of course, the `Sim` object reachable by traversing links in the above definition may also simply be the `Sim` object to which the value belongs. The attribute derivation network defined by the multiple derivation routes has similarities with the network of "design characteristics" used by MacCallum and Duffy's DESIGNER system [MacCallum 1987].

A link value may be:

- UNKNOWN
- NONE
- a pointer to another `Sim` object
- a pointer to an `IAO`

Links provide the basic mechanism for a hierarchical decomposition of the risk model. A `Sim` object may be decomposed into those other `Sim` objects to which it has links (this is generally achieved indirectly using the `ICO` class, see Section 8.5 below). In [Koopman 1995] the author suggests that there are three types of decomposition hierarchy used in design - structural (according to "form"), behavioural (according to "function") and goal-based (according to the needs the design is intended to fulfil). Koopman illustrates that a single design model may combine two, or all three, types of decomposition. In principle, the risk modelling methodology described here could be used to build any of these types of hierarchy, or a single risk model may contain examples of all three. In practice, the case-studies presented in this thesis use structural and functional decomposition strategies.

Multiple derivation routes may be defined for an attribute and in this case they are ordered according to priority. The preferred derivation route is placed first. If it is not possible to obtain a value by this preferred route (because UNKNOWN attributes or links are encountered) then the next derivation route is tried and so on. The first derivation route which can return a value is used in the evaluation and this is termed the *live route.*

A class may have several methods available for calculating a particular attribute value, each of more accuracy than the last, but the most accurate will generally require more information than the least accurate. By using multiple derivation routes, a model can be built which will use the most accurate method for which the input data is available. As information is added to the model, more accurate methods will be used.

Thus it can be seen that the methodology supports the modelling of several types of uncertainty:

- missing attribute values
- missing `Sim` objects
- uncertain attribute values
- design alternatives, only one of which will eventually be chosen
- multiple derivation routes for an attribute value
- heuristic methods (for calculating attribute values) which themselves introduce uncertainty

8.4 Design Stages

The risk model must be able to evolve to reflect the changing state of knowledge about the design. The risk model will become more specific and more detailed as the project proceeds. Previously unknown attributes will be assigned values. Probability distributions for attribute values will become narrower or will be replaced by point values. Objects will be replaced by other objects of more specific class - for example, a generic `Moulding` object may be replaced by an instance of class `InjectionMoulding`. High level assemblies, whose costs were previously obtained using heuristic rules, will be decomposed into their component objects and the costs will be obtained by aggregation.

At each stage in the design process, a new version of the risk model can be stored. The overall risk level in the project can be monitored by comparing the probability distributions obtained for the output attributes at each stage. For example, if the spread in the overall piece part cost for a vehicle area increases from one design stage to the next, this indicates that there is a problem and corrective action should be taken.

8.5 ICO Relationships

In order to decompose an assembly into its sub-assemblies or components as the design proceeds, the risk model must be able to represent the "is-a-component-of" relationship. Relationships between `Sim` objects in the risk model are themselves represented as objects. A relationship between an assembly and its sub-assemblies is an instance derived from a sub-class of the base class `ICO`. There is information which "belongs to" the relationship - for example, the cardinality, the quantity of each sub-assembly - and this is modelled as attributes of the `ICO` object. There are rules for propagating attribute values through the relationship - for example, aggregating costs from the components to obtain the assembly costs - and these are modelled as methods of the `ICO` object. When a designer wishes to decompose an assembly into its components, he or she creates an `ICO` object and creates the component objects and then defines links from the assembly to the `ICO` and from the `ICO` to the components.

The PhysICO class relates physical objects. The class definitions for ICO and PhysICO are shown in Figure 8-2 - the notation used is Fusion class definition syntax see [Coleman 1994]. Notice that the class of the link assembly is specialised in PhysICO from being a generic SimWithID in ICO to being an Assembly in PhysICO. The method piece_cost_fn, for example, aggregates the piece costs for the components and the assembly processes to obtain the piece cost of the assembly. New classes of ICO relationship can be derived, either from PhysICO or directly from ICO. In principle for example, a TaskICO relationship class could be defined which could be used to model the work breakdown structure for a project and which contains the extra methods and data needed for critical path analysis.

```
class ICO isa Sim
 attribute constant assembly : unbound SimWithID
 attribute variable components : bound SimWithID[]
 attribute variable cardinalities : INTEGER[]
endclass

class PhysICO isa ICO
 attribute constant assembly : unbound Assembly
 attribute variable components : bound PhysicalObject[]
 attribute constant piece_cost : FLOAT
 attribute constant logistics_cost : FLOAT
 attribute constant tool_cost : FLOAT

 method piece_cost_fn : FLOAT
 method logistics_cost_fn : FLOAT
 method tool_cost_fn : FLOAT
endclass
```

Figure 8-2: Class definitions for "is-a-component-of" relationships

8.6 IAO Relationships

A risk model may contain several alternatives for a particular Sim object. For example, there may be several alternative assembly processes under consideration for an assembly. Or there may be two different designs, A and B, for the whole assembly under consideration. In this case an IAO (is-alternative-of) object is used to represent the relationship between A and B. Two kinds of IAO object are provided in the base classes : RandomIAO and ByVolIAO. In a RandomIAO, a probability is assigned to each alternative. This represents the perceived likelihood of that alternative eventually being chosen. By contrast, in a ByVolIAO, the choice is not an independent random variable, but depends upon another variable in the model. Here, the choice made between the alternatives is determined by the production volume of a specified PhysicalObject. The class definitions for the IAO classes are illustrated in Figure 8-3. Note that Discrete (which occurs in the class definition for RandomIAO) represents a discrete random variable which is an index into a variable cardinality link/attribute.

```
class IAO isa Editable
 attribute variable alternatives : unbound Editable[]
 attribute variable selection : unbound Editable
endclass

class RandomIAO isa IAO
 attribute constant probabilities : bound Discrete
 method choose_random : Editable
endclass

class ByVolIAO isa IAO
 attribute constant physical_object : unbound PhysicalObject
 attribute variable prod_vols : FLOAT[]
 method choose_by_prod_vol : Editable
endclass
```

Figure 8-3: Class definitions for "is-an-alternative-of" relationships

An instance of an IAO class stores the class of the objects which it relates - it may relate objects of this, or a derived, class. This is termed the *IAO parameter class* (because the class itself is a parameter of the IAO). Links may then be defined to the IAO object from elsewhere in the model - provided that the IAO parameter class is the same as (or derived from) the class of the link.

8.7 Heuristic Methods

Heuristic methods are used to implement "rules of thumb" such as "The piece cost for the trim on a small car with leather trim is usually about 900 pounds and always lies between 850 and 1200 pounds". A heuristic method takes point values as inputs and creates an UncertainValue object (i.e. a probability distribution and its parameters, an uncertain value) as output. So the rule itself introduces uncertainty into the model. Our example rule of thumb could be represented by a heuristic method on class Trim which takes Trim::car_size and Trim::trim_type as inputs and produces a triangular *distribution* with parameters min=850, likely=900 and max=1200 as output when car_size ==small and trim_type==leather. The heuristic method may generate different distributions with different parameters for other values of car_size and trim_type, or indeed it may produce an UNKNOWN value for some cases.

Rules of thumb generally make use either directly or indirectly of historical information - for example the values of 850, 900 and 1200 above. This information can either be hard-coded into the method (stored implicitly) or stored explicitly in the risk model. The advantages of storing the historical data explicitly are that it can be modified without the need for re-compilation and also that it can be browsed by the user.

The example heuristic method above is rule-based. Note that a heuristic method can equally well be used to represent parametric functions for deriving distribution parameters. For example, a heuristic method might take A and B as inputs and create a normal distribution with mean = 34.2 * A + 32.0 and standard deviation = 0.1 * B.

8.8 IVO Relationships

Often, components and assemblies are designed and manufactured in several different variant forms. The "configuration design" - the structure of configurations offered and how they are built from variant parts - has a strong impact on both cost and risk and for this reason, the methodology needs to support modelling of variants.

The decision to offer a new optional feature to the customer has implications for both cost and risk for the design. For example, in a vehicle design program, a design decision to offer the customer a choice of air-bag or none, rather than always including an air-bag, may result in increased piece-part and tool costs[1]. Piece and tool costs are often heavily dependent on production volumes for components and assemblies which in turn are dependent on sales volumes for product offerings. For example, if the designer decides to offer a choice of air-bag or no air-bag, then it becomes necessary to estimate how many of the vehicles sold will include the

[1] This example pre-dates the universal inclusion of driver-side air-bags in all new cars, but is nonetheless helpful for illustrative purposes.

air-bag option, in order to calculate the production volume for those components and assemblies which are only required with an air-bag, and thus determine their piece and tool costs. Additional components and assemblies will have to be designed which will only be used if the air-bag is omitted - for example a plastic central steering-wheel cover. And there may for example be other components and assemblies which are only required if both the air-bag feature and, say, the leather-trim-steering-wheel feature are chosen by the customer. So the combination of features chosen by the customer determines the components required. And thus in general it is necessary to estimate how many vehicles will be sold with each combination of features, in order to calculate the production volumes for components and assemblies.

Not only do sales volumes have a strong impact on cost, but they are also highly uncertain because they are notoriously difficult to predict.

Before describing how the configuration modelling is supported by the methodology, we define some terms. For clarity, the automotive design example is used, however the model generalises easily to other design domains.

Feature

A feature describes the capability of the product in terms of function (see page 2 [Davis 94]). A feature generally has both a price and a cost associated with it.

Product

A vehicle offering. A particular configuration of the vehicle, which a customer may purchase. There will be a limited number of products offered - these are the "brochure models" or standard offerings.

Customer Option

An extra feature or set of features which the customer may choose to purchase. Each product has a set of valid customer options which are available - the customer may choose whether or not to purchase each option. Thus there are generally a very large number of possible configurations for the vehicle, obtained by considering all the possible combinations of products and customer options. For example, 10 products each with 10 valid customer options gives $10 \times 2^{10} = 10{,}240$ possible vehicle configurations. Customer options are assumed to be independent. What is meant by independence in this context is that no physical object provides more than one customer option for a particular product. Consequently, it is adequate to specify the sales volume for each customer option independently in order to calculate the production volumes for the physical objects (see Section 9.3.2 of Chapter 9) - it is not necessary to specify the sales volume for each possible combination of customer options.

Generic PhysicalObject

A part which has more than one variant included in the model.

Variant

A Sim object (usually a PhysicalObject) which belongs to a related set of variants. The definition of such a set is that they will all be manufactured but a maximum of one member of the set will be included in any one particular product instance. Also, all the members of a set of variants fit into the Bill-Of-Materials hierarchy in the same place.

There are four basic classes which are used in the configuration model of a design (all described in detail in Appendix D): The Product is a particular configuration which will be offered to the customer. A Feature is a property of a product which usually has a price associated with it; the customer will be willing to pay more for a product with more features. Examples of features in a vehicle design might include "passenger air-bag", "sporty-trim", "anti-lock brakes" or "sunroof". And PhysicalObjects are the components or assemblies which provide the features. A Product will also have a set of optional CustomerOptions which will be offered with it. The customer will choose a Product and any combination of its CustomerOptions. A product will have many valid customer options but a customer option is identified with a particular product. A customer option may contain many features and a feature may appear on many customer options. These relationships are illustrated in Figure 8-4 (this is a simplified form of the Fusion object model).

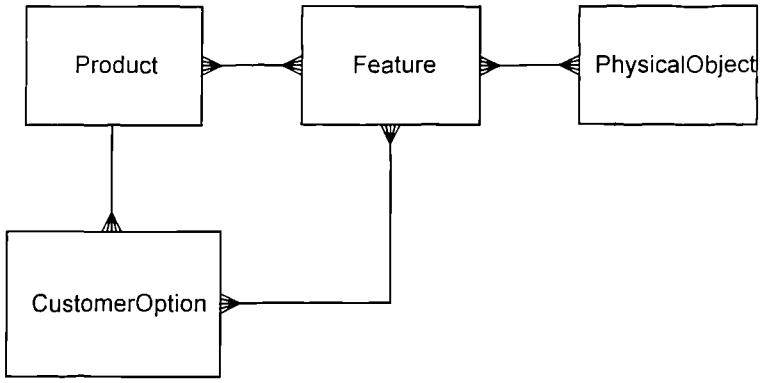


Figure 8-4: ER diagram illustrating the relationship between the component model and the feature model

A risk model with variants will include a model of the PhysicalObjects required, related by ICO (is-a-component-of), IVO (is-a-variant-of) and IAO (is-alternative-of) relationships, along with their ManufacturingProcesses, Materials etc. This is termed the ComponentModel in what follows. A separate area of the risk model contains definitions for the products which will be offered, defined by the features they will provide and the customer options which will be valid for them. This is termed the FeatureModel. The feature model contains the necessary information to "switch-in" or "switch-out" sections of the component model depending upon the features which are provided by the product of interest and the chosen customer options. Features can select between variant PhysicalObjects - i.e. between PhysicalObjects which are related by an IVO. Thus the component model will contain representations of all the variant PhysicalObjects which are in the design.

At the highest level the feature model consists of a list of products and a link to the current product of interest - the selected product. Each product contains a list of links to the features which are provided by it and also a list of links to the customer options which are valid for it. A feature contains optional links to

other features which it "implies" - features which must also be present or which can be considered as its sub-features - as well as information about which parts of the component model should be selected if this feature is present. The feature model is illustrated using an entity-relationship diagram in Figure 8-5.

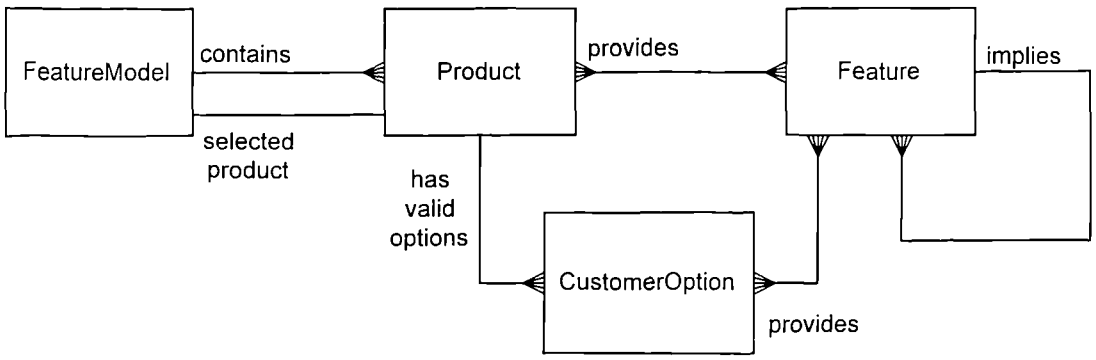


Figure 8-5: ER diagram illustrating the feature model

8.9 Summary

An object-oriented modelling methodology has been described which allows the incorporation of probabilistic uncertainty into the model in several respects. An uncertain value may be defined for any object *attribute* by specifying a probability distribution and its parameters - this is achieved using an object of class `UncertainValue`. An uncertain value may also be defined for any *link* between objects by specifying a set of possible destination objects for the link along with a method for making a selection between them, where the method may return an uncertain value. This is achieved using an object of class `IAO` (is-an-alternative-of) - one of three categories of relationship object, the other two being `ICO` (is-a-component-of) and `IVO` (is-a-variant-of). A *method* which returns a value for an attribute may also introduce uncertainty into the model - in the methodology, such a method is termed a *heuristic method* and takes point values as its inputs but generates an `UncertainValue` as its output.

Both attributes and links may be assigned an explicit value of UNKNOWN, if no information is available. A sequence of evaluation methods (termed *derivation routes*) may be defined for a numerical attribute value. The derivation routes are placed in an order of preference, with the most accurate route, typically requiring the most input information, first, and the least accurate, requiring the least information, last. The route used in an evaluation of the model will be the most accurate derivation route which can be evaluated without encountering any UNKNOWN attribute values or links. In this way, the methodology supports the evaluation of incomplete models.

The `IVO` (is-a-variant-of) relationship is used to model sets of variant parts - sets of `PhysicalObjects`, only one of which will be included in a particular configuration of the designed artefact. The risk model is divided into a `FeatureModel`, describing the `Products` and `CustomerOptions` which will be offered to the customer, in terms of `Features`, and a `ComponentModel` which contains the components and assemblies used to provide the features. The component model includes all the variant `PhysicalObjects`, along with their materials, manufacturing processes etc., structured by `ICO`, `IAO` and `IVO` relationships. The information in the feature model can be used to "switch in and out" parts of the component model so that it

contains only those components and assemblies required to build a specified product with specified customer options - i.e. a specified configuration of the designed artefact. In this way the methodology supports the modelling of the configuration design of the artefact.

It is important to note that, in common with many object-oriented modelling systems, the methodology described uses two different and complementary hierarchical network decompositions. The *inheritance hierarchy*, representing the "is-a-type-of" relationship, relates classes. The network of *links* between objects form the basis for the `ICO`, `IAO` and `IVO` relationships - links relate objects (i.e. instances of classes) and may generally be regarded as representing a low-level "is-a-part-of" relationship. The Bill-of-Materials hierarchy (defined by the `ICO` relationships) provides a simplified high-level view of the link network. A third network also exists within the risk model; the attribute derivation network is defined by the derivation routes. It should be noted that although the linked network of objects provides a storage place for the attribute values, neither the inheritance hierarchy nor the link network define the attribute derivation network.

The methodology presented requires that the leaf nodes in the attribute derivation network, i.e. the `UncertainValue` objects, should be independent of each other. This decision was taken because it was felt that the intended users of the methodology would have difficulty in specifying the statistical dependency between such variables. It was felt that where such correlations do exist, the attribute derivation network should be further extended to model the cause of the statistical dependency as a causal dependency between the correlated attributes and a common lower level attribute value - with the lower level attribute values being independent. The danger with this approach is that the user will ignore known statistical dependencies because of the additional effort required to further decompose the model - particularly when the cause/s of the statistical dependency are complex and difficult to model or even unknown. There is no limit to the potential inaccuracy which may be introduced into the risk model by ignoring statistical dependencies, and this is an important limitation of the methodology as presented.

Chapter 9 presents the mathematical and computational techniques required to implement the methodology and Chapter 10 describes the software tool which has been developed to do so. In Chapter 11, some examples are presented which illustrate the use of the methodology.

8.10 References

[Davis 1994] Davis, N. "Model:ProdDefinition", *STARTED report /STARTED/ROV 003/1*, June 1994.

[Koopman 1995] Koopman, P. J., "A Taxonomy of Decomposition Strategies Based on Structures, Behaviours and Goals", *Proc. Design Engineering Technical Conferences*, ASME, Boston,, vol. 2, pp. 611-618, 1995.

[Lehane 1990] Lehane, K. J., "*Computer Aids for Variant Design*", PhD Thesis, University of Bristol, 1990.

[MacCallum 1987] MacCallum, K. J. and Duffy, A., "*An Expert System for Preliminary Numerical Design Modelling*", Design Studies, vol. 8, no. 4, pp. 231-237, October. 1987.

CHAPTER 9

Methods of Evaluating Risk

This chapter describes in detail the mathematical and computational techniques used in the risk tool. The first section describes the sampling techniques used during Monte Carlo simulation of a risk model. The second section presents the method adopted for performing a risk sensitivity analysis on the risk model - identifying which are the major sources of uncertainty in a risk model, in a hierarchical and object-oriented fashion. The issue of hierarchical decomposition, both of the simulation itself and of the risk sensitivity analysis, is discussed here. In the third section, the approach adopted to support configuration modelling under uncertainty is described; several different variants of an assembly or component may be designed, each of which will only be used in certain configurations of the final product. In the third section, a model which has been developed to represent such variants, and also the uncertainty in product sales volumes and hence component production volumes, is presented.

9.1 Simulation

In this section, the technique and the algorithms used to propagate uncertainty in attribute values through the risk model are described. The reasons for choosing Monte Carlo simulation as a means of evaluating the risk model are presented, and the sampling strategies chosen (Latin hypercube sampling and sampling without replacement for integer values) are described as well as the basic random number generator. The implementation of the algorithms in the risk modelling tool is presented in Chapter 10 and Appendix K, where the object interaction algorithms used in a Monte Carlo simulation are described.

9.1.1 Why Monte Carlo?

Using the methodology presented in Chapter 8, an uncertain attribute value may be represented in the risk model using a probability distribution function and its parameters. Attribute values are combined using methods (which are functions, associated with a particular class, which return values), to yield the values of other attributes in the model. It was therefore necessary to choose one or more techniques to propagate the uncertainty through the methods. As mentioned in Chapter 6, one of the key concepts in object-oriented programming is the encapsulation of a method's implementation within its class definition; thus, the implementation of a method can be modified in ways unforeseen by the original programmer, without necessarily requiring any changes to be made elsewhere in the program. In order to encapsulate the methods in this way, it was important that at least one of the techniques chosen for propagating uncertainty should not place any constraints on the form of the method. In this way, the risk tool could be used with arbitrarily defined classes of Sim object, with arbitrary "black-box" methods; methods which may be non-linear, non-monotonic or even discontinuous. Similarly, it was required that the tool could provide arbitrary probability

distribution functions, in order that new functions could be added at a later date without requiring other changes to the risk modelling tool. Another important consideration in choosing the uncertainty propagation technique was that the method chosen should not introduce any bias (i.e. systematic error) into the results.

There are broadly three types of computational technique available for propagating probability distributions through a combining function:- analytical methods, numerical methods or simulation. In Appendix B, a summary is given of simulation and also of the main numerical methods - numerical convolution, discrete probability distributions and histogram-based methods. Several examples of analytical methods which can be applied in particular circumstances are also presented - for example, level 1, 2 and 3 reliability methods or tolerancing methods. However, analytical methods cannot be used with arbitrary combining functions or arbitrary PDFs - indeed, most rely on assumptions of linearity for the function and a normal distribution for the PDFs. Considering the numerical techniques, numerical convolution can be performed using numerical integration, but still requires knowledge of a unique differentiable inverse for the combining function, and the derivative of this inverse. Thus numerical convolution cannot be used where the form of the combining function is arbitrary. The discrete probability distribution (DPD) approach involves an inherent bias - for example yielding too low a variance for the output distribution if the input distribution is uniform. Histogram-based methods also cause a systematic error. The controlled-interval histogram-based methods can reduce the error to an arbitrary level, but require prior knowledge of the form of the function and the PDF in order to do so.

It was concluded that the only technique which can be used to iteratively propagate arbitrary PDFs through arbitrary combining functions without systematic error is Monte Carlo simulation. The sampling error which is involved in simulation can be reduced to an arbitary level by increasing the number of iterations - the lack of systematic error means that over several simulations (using different seed values), the average value for a statistic of the output distribution will tend towards the correct value. Monte Carlo simulation can be computationally expensive, but increased computing power during the past decade has meant that this has become a far less important consideration than in the past. It seems likely that the amount of computing power available will continue to increase in the foreseeable future.

The idea of Monte Carlo simulation is well-known (and is presented in Appendix B), but is summarised here and the notation used in the remainder of this Chapter is introduced. Suppose we have N independent random variables, $\boldsymbol{X}_1,...,\boldsymbol{X}_N$, whose probability density functions (PDFs) are known and are $p_1(x_1)$, ...,$p_N(x_N)$ respectively. We wish to calculate the PDF $p_Y(y)$ of the random variable $\boldsymbol{Y}$, where $y = f(x_1,...,x_N)$ and $f()$ is an arbitrary function. In a Monte Carlo simulation, a sample of M values is generated for each x_i, according to the known PDF $p_i(x_i)$, where M is a large number. The function $f()$ is evaluated on each of the M sample values for the vector $<x_1,...,x_N>$, to produce a set of M sample values of variable y. The distribution of this sample of y values is determined, and is an approximation to the PDF of $\boldsymbol{Y}$. The larger the value of M, the better the approximation.

It is also possible to perform a Monte Carlo simulation when the $\boldsymbol{X}_1,...,\boldsymbol{X}_N$ are statistically dependent, by sampling $<x_1,...,x_N>$ according to the joint PDF of $<\boldsymbol{X}_1,...,\boldsymbol{X}_N>$, but this was not implemented in the risk tool and is thus not discussed further here. It was decided that the specification of joint PDFs was too distracting a

task for the intended users of the tool, and that the existence of statistical dependencies between attribute values should be broken down into a causal dependency between lower-level, independent, attribute values. It was felt that this would increase the legibility of the risk model.

In order to perform a Monte Carlo simulation, a technique is required for generating sample values according to a known PDF, and this is addressed in the next two sections.

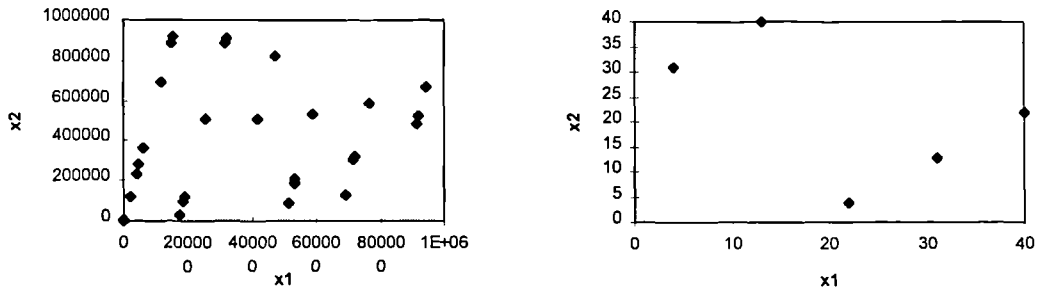
9.1.2 Pseudo-Random Number Generators

All computational techniques for generating samples of random variables require a pseudo-random number generator. A pseudo-random number can be regarded as a sample value from a uniform (i.e. rectangular) PDF over a known interval [0, *Max*]. Pseudo-random integer values are generated and are then processed in some way which is specific to the required PDF and the sampling strategy.

The simplest way to generate a sequence of pseudo-random integer values v_1, v_2,... is to repeatedly use a *linear congruential generator*:

$$v_i = (Av_{i-1} + C) \text{ modulo } (Max + 1)$$

with suitably chosen values for the constants v_0, A and C. This is the technique used by most system-provided functions such as the *rand()* and *srand()* functions provided as part of the ANSI and UNIX standards for the 'C' programming language. However, it can be seen that if random values are sampled for a large number of variables $x_1,..,x_N$ sequentially, where N is a significant proportion of *Max*, then the values generated will display sequential correlations (see [Press 1986], for example) - in other words, the resultant points in N dimensional space will not be evenly scattered, but will lie in $N-1$ dimensional strata. For two variables x_1 and x_2, for example, if *Max* = 45, the resultant points would lie on lines instead of being evenly scattered in the plane when *Max* = 1000,000 as shown in Figure 9-1.



$A = 6$ $A = 6$

$C = 97$ $C = 97$

Max = 1000,000 *Max* = 45

v0 = 13 *v0* = 13

Figure 9-1: 30 sample values illustrate sequential correlations in a linear congruential generator

The value of *Max* used in system-provided functions is determined by the number of bytes used to store an integer - for the Microsoft C++ compiler under Windows 3.1 for example, it is 32767, which is 7FFF in hexadecimal, since 2 bytes are used to store an integer. Careful choice of the values *A* and *C* can increase the number of strata, but (see [Press 1986]) the number of strata can never be more than approximately:

$$(Max+1)^{N}$$

To overcome the problem of sequential correlations, a linear congruential generator (LCG) can be used with the shuffling procedure developed by Bays and Durham (see [Knuth 1981], sections 3.2 and 3.3). A table of *K* random integers is initialised and an initial random number *I*, is chosen from within [0, *Max*]. The following algorithm is then repeatedly performed:

1. Set *J* = INTEGER(*I* * *K* / *Max*)
2. Set *Out* = element number *J*, from the table.
3. Set *I* = *Out*.
4. Store a new pseudo-random integer, generated using the LCG, into element number *J* in the table.
5. Return *Out* as the next pseudo-random output.

A further improvement can be made by employing three LCGs, *ran1()*, *ran2()* and *ran3()*, in a similar shuffling routine, each using different values of *Max*, *A* and *C*. The first LCG generates the most significant part of the output value, the second LCG generates the least significant part and the third generates the index into the shuffling table. The table is initialised using *ran1()* and *ran2()*. The following algorithm is presented in [Press 1986] and attributed to Knuth:

1. Set *I1* = *ran1()*
2. Set *I2* =*ran2()*
3. Set *I3* = *ran3()*
4. Set *Out* = element number INTEGER(*I 3* * *K* / *Max3*), from the table.
5. Construct a new pseudo-random integer from *I1* and *I2* and store it into element INTEGER(*I 3** *K* / *Max3*) of the table.
6. Return *Out* as the next pseudo-random output.

The values chosen for *Max*, *A* and *C* for any LCG must satisfy several conditions. Clearly *Max* * *A* must be sufficiently small to avoid integer overflow and *C* should not be a divisor of *Max*. It is also clear that some combinations of values will never generate all possible output values, and thus should not be used. Subject to these constraints, *Max* should be chosen as large as possible. Some sets of known "good" values are published in [Press 1986].

Given an algorithm which can generate "good" pseudo-random sequences of integers on some fixed interval [0, *Max*], the result can then be divided by *Max* to simulate a uniformly distributed random variable on [0,1].

Random variables with other PDFs can then be simulated from this unit uniform rv, using the techniques described in Section 9.1.3.

The algorithm presented above is considerably more computationally expensive than using a system-supplied function, but has several advantages. As well as avoiding sequential correlations and having a finer resolution than such functions, by avoiding system-supplied functions it is portable across different platforms - and the results obtained do not depend upon the number of bytes used to store an integer in one particular system.

9.1.3 Sampling Strategies

In this section, when the term "random number" is used, it should be understood to mean a pseudo-random number generated using a computational algorithm. References are also made to "generating random variables" and this should be understood to mean generating a sequence of pseudo-random numbers which are distributed approximately according to the PDF of the specified random variable.

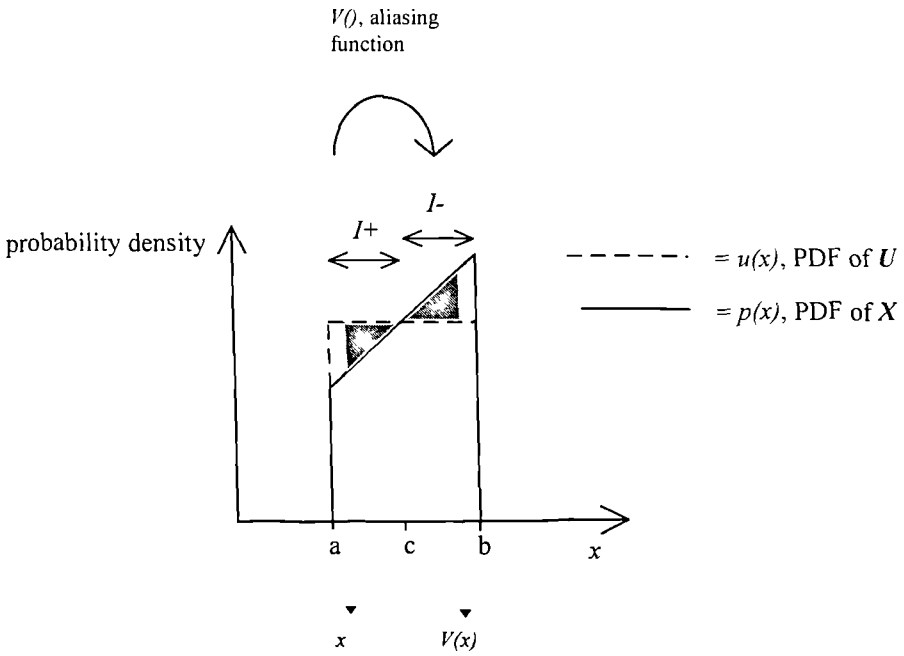
One method for generating a discrete or continuous random variable X of known PDF $p(x)$ is the rejection method, which is described in Appendix B. This method may be used if $p(x)$ is known in analytical form or if it is only known in numerical form. Uniformly distributed sample values are generated over the domain of $p(x)$, but only a subset are retained as part of the sample. The proportion which are retained is determined by the required distribution. The decision whether to accept or reject a sample value x is made by generating a sample value u from a second random variable, U, with unit uniform distribution. If $u <= p(x) / a$, (where a is the peak, or maximum value, of $p()$) then the sample is accepted, otherwise it is rejected. The algorithm can be expressed in pseudo-code:

```
DO
        generate a random value u1 from U
        x = B · u1 + C
        generate a random value  u from U
WHILE  p(x) / a <  u
RETURN x
```

B and C are constants which map the unit interval [0, 1] onto the domain of $p(x)$. Distribution-specific variations on the rejection method use any distribution $h()$ which has the property that $p(x) < a \cdot h(x)$, for some constant value a, and has the same domain as $p()$, to generate the initial sample value. But one can always choose $h()$ to be a uniform distribution and a to be the peak value of $p(x)$, as above, regardless of the distribution $p()$.

This method is inefficient, since random numbers are generated and then rejected. In a bell-shaped distribution, for example, very many random numbers will be rejected from the "tails" of the distribution (i.e. near to the extreme values, where $p(x)$ is usually small), and the central portion of the distribution will be built up more rapidly than the tails. The variation proposed in [Walker 1977] and extended to continuous

variables in [Deák 1981] avoids the speed penalty of generating values which are then rejected by using an *aliasing function*. It can be seen from the pseudo-code above that some of the values generated for x are very likely to be rejected, whereas others are very likely to be accepted. The values of x which are more likely to be rejected are termed the excess values and are said to lie in the excess area $I+$, and those which are less likely to be rejected are termed shortage values and are said to lie in the shortage area $I-$. Figure 9-2 shows an example where a random variable with a trapezoid distribution is required. An *aliasing function*, $V()$, maps $I+$ onto $I-$, so that instead of discarding the rejected values in $I+$, they are transformed into values in $I-$ which can be accepted. In the example shown in Figure 9-2, $u(x)$ is a uniform distribution, and $p(x)$ a trapezoid:



$I+$ = excess area

$I-$ = shortage area

Figure 9-2: Using an aliasing function to generate samples with a trapezoid PDF

$$u(x) = \begin{cases} 1 / (b - a) & \text{if } x \in [a, b] \\ 0 & \text{otherwise} \end{cases}$$

$$p(x) = \begin{cases} Mx + D & \text{if } x \in [a, b] \\ 0 & \text{otherwise} \end{cases}$$

We want to define a (linear) aliasing function V: $I+ \rightarrow I-$ which satisfies:

$$\int_x^c \{u(y) - p(y)\}dy = \int_c^{V(x)} \{p(y) - u(y)\}dy$$

for all x. The coefficients for a linear $V()$ can be determined from this equation, and the modified rejection method algorithm can then be expressed in pseudo-code thus:

```
generate uniform random number u1 on [a, b]
IF u1 ∈ I- THEN
        RETURN u1
ELSE
        generate uniform random number u on [0, 1] (i.e. from U)
        IF (b - a) * p(u1) > u THEN
                RETURN u1
        ELSE
                RETURN V(u1)
        ENDIF
ENDIF
```

Although the use of an aliasing function improves the computational efficiency of the rejection method, the inverse cumulative distribution method is considerably more efficient. The inverse cumulative distribution method, shown in Figure 9-3, may however only be used if the PDF, $p()$, is known in analytical form and if a unique inverse cumulative distribution function can be constructed.

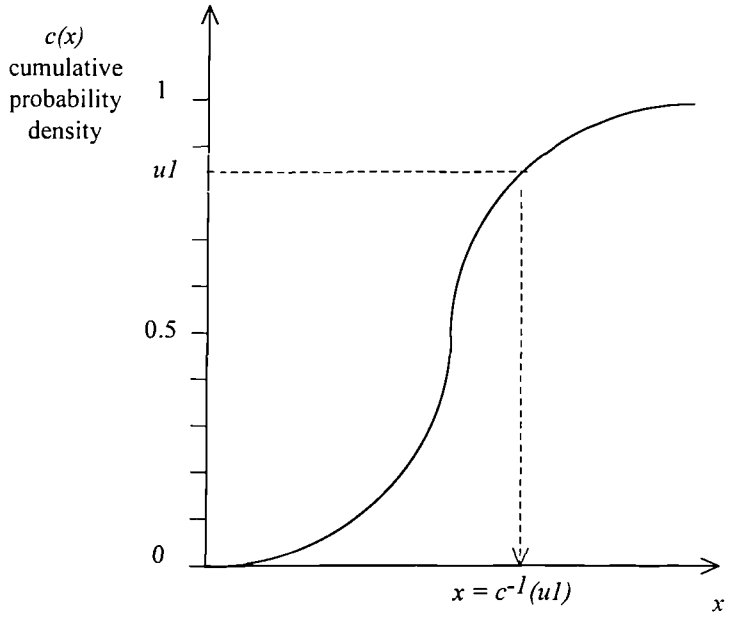


Figure 9-3: The inverse cumulative distribution method

The cumulative probability density function, or CDF, is defined by:

$$c(x) = \int_0^x p(s)ds$$

Thus, $c(x)$ is always a monotonically increasing function taking values in the interval [0,1]. In the inverse cumulative distribution method, a random number $u1$ is generated from the random variable $\boldsymbol{U}$ with unit uniform distribution. This value is interpreted as a value of $c(x)$, and projected onto the x-axis to yield the corresponding x-value, $c^{-1}(u1)$, which is returned.

In recent years, techniques known as *stratified sampling* and *Latin hypercube sampling*, have been widely used to improve on the efficiency of basic Monte Carlo simulation. These are known as variance reduction techniques, because they reduce the variance between the expected value of some parameter (e.g. mean, or other moment) of the output random variable and its estimate obtained from the sample. The output distribution can be accurately reconstructed from a smaller number of sample values when using variance reduction techniques than when using the rejection method. Thus, even though it may require more computation time to generate each individual sample value using variance reduction than rejection, fewer sample values are needed to build up the output distribution and thus variance reduction techniques are considerably more computationally efficient overall.

The Latin hypercube sampling method [Iman 1980] is based on the idea, illustrated in Figure 9-4, of dividing the vertical axis of a graph of the cumulative probability into a fixed number of equal strata and then taking an equal number of sample values from each stratum.

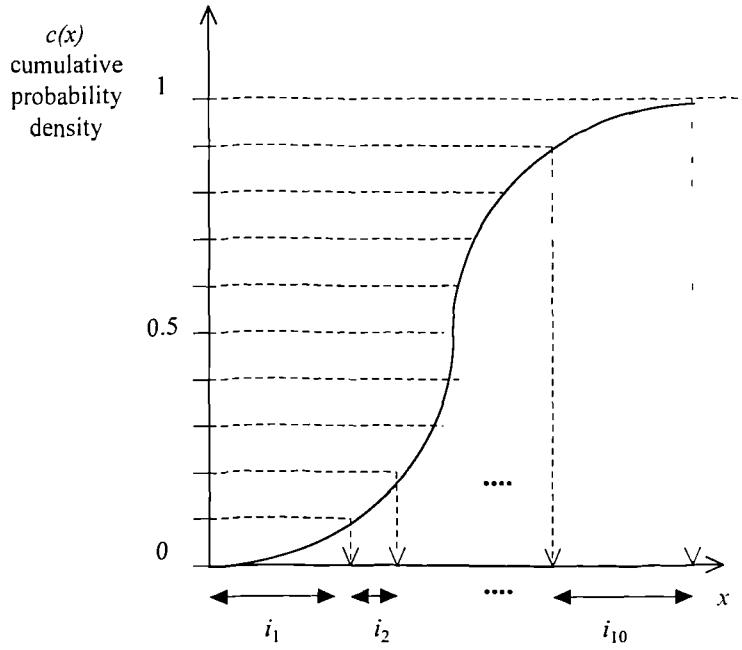


Figure 9-4: Latin hypercube sampling with 10 strata

As shown in Figure 9-4, *Num* equal divisions on the vertical axis divide the x-axis into a set of unequal intervals $i_1,...,i_{Num}$. A stratum (say the j'th) is chosen at random, using a random generator for U, and a random value is then generated within the x-axis interval i_j for that stratum. A record is kept of how many samples have been generated from each stratum and an equal number of sample values are taken from each interval i_j. The maximum variance reduction can be achieved by taking exactly *one sample from each* i_j - thus the number of strata is equal to the number of iterations in the simulation. Definition of a suitable algorithm to implement this method was found to be fairly straightforward. The algorithm can be expressed in pseudo-code thus:

NumIts	number of iterations in Monte Carlo simulation
SamplesLeft	number of sample values left to generate (initialised to *NumIts* at start of Monte Carlo simulation)
CDFInverse()	the inverse CDF

```
generate uniform random number u1 on [0, 1] (i.e. from U)
X = INTEGER(SamplesLeft * u1)
SamplesLeft = SamplesLeft - 1
J = 0
Sum = -1
WHILE Sum <> X
        IF J'th stratum has not yet been sampled THEN Sum = Sum + 1
        J = J + 1
ENDWHILE
J = J - 1
record that J'th stratum has now been sampled
generate uniform random number u2 from U
u2 = (u2 + J) / NumIts
RETURN CDFInverse(u2)
```

Since only one sample is taken from each stratum, a record of the number of samples taken can be stored as an array of *NumIts* one-bit values; for clarity however, direct reference to this bit-array is omitted from the pseudo-code above. The X'th stratum amongst those left unsampled is chosen at random, which is the J'th stratum in total. A value $u2$ is then chosen at random from within the J'th interval on the cumulative probability axis, and then $u2$ is projected onto the x-axis (using the inverse CDF) to yield the sampled value.

It can be seen from the above that Latin hypercube sampling is only applicable to continuous random variables. There is a memory overhead incurred of one bit per iteration per random variable, and also a function must be implemented which returns the inverse of the CDF for the desired distribution.

Latin hypercube sampling is sometimes referred to as an example of a *stratified sampling* technique, however here we use the term *stratified sampling* to indicate a slightly different, alternative, sampling strategy which is applicable when a sample is required for a vector of random variables, $<\boldsymbol{X}_1,...,\boldsymbol{X}_N>$. The cumulative probability space for the whole vector is partitioned into a fixed number of equal-sized rectangular blocks in N-dimensional space, rather than partitioning each axis independently. Stratified sampling can be extended to *proportional allocation* with one observation taken per stratum of the sample space. For N independent rvs, the cumulative PDF of each rv is divided into S strata of equal probability, where S^N = *NumIts*, and a single sample taken from each. In [McKay 1979], Monte Carlo sampling is compared with stratified sampling and the Latin hypercube technique and it is shown that Latin hypercube sampling gives better variance reduction than stratified sampling with one observation taken per stratum for uniformly distributed rvs.

Whether stratified sampling or Latin hypercube sampling will yield a more accurate representation of the distribution of $y = f(x_1,...,x_N)$ in a Monte Carlo simulation of, say, 100 iterations, depends upon the nature of the function $f()$ and the distributions of the x_i. For example, suppose $f()$ is linear, N=2, and x_1 has a larger coefficient than x_2 in $f()$ and a wider multi-modal (multi-peaked) PDF. In such a case, Latin hypercube

simulation will almost certainly yield better results, because the x_1 axis is divided into 100 strata and all the multiple peaks are more likely to be represented than with stratified sampling, where the x_1 axis would only be divided into 10 strata and a small peak might be omitted. The fact that the different combinations of x_1 and x_2 have not been evenly sampled is not so significant because x_1 dominates - it is more important in such an example that the different possible values of x_1 should be represented.

A decision was taken to use Latin hypercube sampling for the risk tool: it could not be predicted whether this would yield "better" or "worse" samples than using proportional allocation, but it was felt that the additional complexity of a proportional allocation algorithm could not be justified, at least initially. The Latin hypercube algorithm given above is not applicable to generation of samples for *discrete* random variables. The algorithm chosen for discrete rvs is described in the following section.

9.1.4 Sampling Without Replacement

It could be argued that the equivalent variance reduction technique to Latin hypercube sampling for discrete random variables is *sampling without replacement*. Conceptually, this can be imagined as placing *NumIts* chips, each marked with a discrete value and where the quantity of chips with a given discrete value is determined by the desired PDF, into a bag and then making *NumIts* selections from the bag at random. After each selection, the counter is discarded and is not replaced in the bag. Sampling without replacement was chosen as the algorithm for generating discrete random variables in the risk tool.

Implementing this method requires that a record is kept during simulation of the number of sample values which have been generated with each discrete value - effectively, the number of chips left in the bag. An algorithm to implement the method is as follows:

NumDiscreteValues	the number of discrete values in the required PDF
SamplesLeft	number of sample values left to generate (initialised to *NumIts* at start of Monte Carlo simulation)
DiscreteValues[ ]	is an array containing the discrete values (may be integer, floating point or Boolean)
Probs[ ]	is an array of floating point values, containing the probability associated with each discrete value
ChipsLeft[ ]	is an array containing *NumDiscreteValues* integers, which is initialised thus:

```
FOR i = 0 TO NumDiscreteValues - 1
        ChipsLeft[i] = INTEGER( Probs[i] * NumIts )
NEXT i
```

Each sample value is generated thus:

```
generate uniform random number u on [0, 1] (i.e. from U)
WhichChip = INTEGER( u * SamplesLeft )
Sum = 0
FOR i = 0 TO NumDiscreteValues
        Sum = Sum + ChipsLeft[i]
        IF WhichChip < Sum THEN
                ChipsLeft[i] = ChipsLeft[i] - 1
                SamplesLeft = SamplesLeft - 1
                RETURN DiscreteValues[i] and HALT
        ENDIF
NEXT i
```

This concludes the description of the algorithms used during simulation of the risk model.

9.2 Risk Sensitivity Analysis

Algorithms have been developed for identifying which are the major sources of uncertainty in a risk model, in a hierarchical and object-oriented fashion - in this section, the algorithms are described. This is termed *risk sensitivity analysis*. In the first sub-section, the problem is formulated without considering the object-structure of the model. The second sub-section describes the available numerical techniques and their applicability to the problem is compared. In the third sub-section, the problem is re-formulated taking into account the object structure of the model. The fourth and final sub-section describes the solution adopted. The algorithms described were not fully implemented in the risk tool, and risk sensitivity analysis was not incorporated into the user-interface, but the algorithms were evaluated on the Rover case study using the risk tool and some additional calculations which were performed in simple spreadsheets. The Rover case study is described in Chapter 7, Section 3 and the implementation of the risk sensitivity analysis algorithm in Chapter 11, Section 3.

The solution proposed makes use of stored samples (of random variables) which are generated during partial simulations of the model. This reduces the additional cost (in terms of computation time and memory) of performing a risk sensitivity analysis over and above the cost of performing a simulation. These samples are stored to disk and can be re-used in a simulation, thus also reducing the computational cost of a simulation following small localised changes to the risk model. The hierarchical nature of the model is intrinsic to the algorithms described. The following terms are introduced in this section:

sensitivity

> The sensitivity of the model output Y to an input x is a measure of the size of the change in Y which would result from a unit change in x, about the current value of x. In a stochastic model, the sensitivity must be defined as the expected value of the change in Y which would result from a unit change in x.

risk sensitivity(RS)

> The risk sensitivity of Y to x is intended to provide a measure of the amount of the observed variance in Y which is "due to" variance of x; thus the risk sensitivity is a combination of the sensitivity of Y to x and the variance of x. More precisely, it is hoped to provide some measure of the reduction in output variance which could be expected if x were to be replaced by a point value. The risk sensitivity is thus defined as the expected value of the reduction in variance of Y when x is replaced by a point value.

interaction effect

> An interaction effect exists between two inputs xi and xj when the effect of xi upon Y depends upon the value of xj. This is an example of a two-way interaction effect. In a three way interaction effect, the effect of xi depends on the values of both xj and xk, and so on. An interaction effect arises, for example, from the cross-term a_{12} in a model of the form:

$$Y = a_1 \cdot x1 + a_2 \cdot x2 + a_{12} \cdot x1 \cdot x2$$

container object

A Sim object which is a node in a bill-of-materials type decomposition of the risk model. Typically, assemblies and components.

interface attribute

An attribute which belongs (directly or indirectly) to one container object and is also used by one or more other container objects.

9.2.1 Simplified Problem Formulation

9.2.1.1 Definition of Risk Sensitivity

Let ***Y*** be the output attribute (a random variable). If we initially ignore the hierarchical nature of the model (its object-structure) and simply consider the attribute dependencies for the whole model, then we can express ***Y*** as a function *F*, of *n* independent input random variables ***x1,..,xn***:

$$\boldsymbol{Y} = F(\boldsymbol{x1},\ldots,\boldsymbol{xn})$$

where each input random variable ***xi*** may be either:

- a *leaf rv*: a uncertain-valued attribute defined directly by the user (using `#uncertain` in the Sim file),
- or a *heuristic rv*: a random variable created by a heuristic method,
- or an *IAO rv*: a discrete integer-valued random variable used to choose between alternatives in a Random IAO.

The important distinction here is between *numerical* variables (leaf and heuristic rvs) and *categorical* variables - an IAO rv takes a value which has neither size nor meaningful order, it merely identifies a category. We do not have a closed form for *F*, because of the use of "black-box" methods in the Sim model. *F* may well be discontinuous and non-monotonic. As a minimum requirement, we wish to rank the input variables according to the contribution which they make to the variance in the output, and identify the major contributors. Preferably, we would also like to provide an estimate of the risk sensitivity *si* for each input variable ***xi***; where *si* is some measure of the contribution made by ***xi*** to the variance in the output. To be more precise, we can define the risk sensitivity (RS) to be the expected value of the variance reduction in the output which would be achieved if we were to replace ***xi*** by a point value. This definition of RS is meaningful when applied to leaf, heuristic or IAO rvs and as a statistic it has an obvious meaning.

If ***xi*** were to be replaced by a point value, $\boldsymbol{xi} = X$, and all the other risk model attributes were left unchanged then the resultant variance of the conditional PDF for ***Y*** given ***xi*** will in general depend upon *X*, according to some function *V*:

$$\mathrm{var}(\boldsymbol{Y} \mid \boldsymbol{xi} = X) = V(X)$$

What is meant by the definition of RS given above is that the RS is then given by the expected value of the random variable $V(\boldsymbol{xi})$, subtracted from the original variance of $\boldsymbol{Y}$:

$$si = \text{var}(\boldsymbol{Y}) - \text{E}(V(\boldsymbol{xi}))$$

9.2.1.2 Direct Calculation Methods

If we have an IAO rv with k alternatives with probabilities $\{p1...pk\}$, then si can in principle be calculated directly (if somewhat expensively) by carrying out k sub-simulations, each with a particular alternative selected. If the variance of $\boldsymbol{Y}$ in the j'th sub-simulation is vj then the expected value of the variance is given by the weighted average and so:

$$si = \text{var}(\boldsymbol{Y}) - \Sigma pj.vj$$

The computational expense of this method may be acceptable, provided that the number of alternatives is relatively small (which it generally will be) and provided that the number of iterations per sub-simulation is also kept small compared to the number used in a full simulation. Another way to reduce the computational expense is to re-use values of $\boldsymbol{Y}$ which had previously been stored during an ordinary simulation (this would mean also storing samples for each IAO rv and using them to partition the $\boldsymbol{Y}$ sample, which would be reasonable[1]). However, the equivalent "direct calculation" method for numeric-valued rvs (i.e. leaf and heuristic rvs) would be even more computationally expensive. For a single numeric-valued rv, the direct calculation method would involve performing K sub-simulations, where K is a reasonable number (see Improvement 1, below), each with a different fixed value of $\boldsymbol{xi}$ and storing the variance obtained in $\boldsymbol{Y} \mid \boldsymbol{xi}$ for each sub-simulation. The value of $\boldsymbol{xi}$ used in each of the sub-simulations would be given by $\boldsymbol{xi}$'s PDF. And si would then be given by the mean of all the variances calculated. This method would therefore involve a kind of "simulation of simulations" for each $\boldsymbol{xi}$. Fortunately, there are several possible improvements on the direct calculation method. Two such improvements are presented below:

9.2.1.3 Improvement 1: approximation by a discrete PDF

K (the number of sub-simulations), can reasonably be made very small - say 3 - if the distribution of $\boldsymbol{xi}$ is approximated by a discrete distribution. Such an approximation can be considered as making a step-function approximation to the cumulative PDF of $\boldsymbol{xi}$. Suppose, for example, that we want to approximate a general continuous PDF f(x) of $\boldsymbol{xi}$ with cumulative density function (CDF), c(x) by a discrete PDF which has 3 discrete values $d1 < d2 < d3$ with probabilities of $p1$ = 10%, $p2$ = 80% and $p3$ = 10% respectively. We want to determine $d1$, $d2$ and $d3$. We know that there is a 10% probability that x lies within the band b1 (shown in Figure 9-5) thus the value we choose for $d1$ should lie somewhere within b1. Similarly, $d2$ should lie in b2 and $d3$ should lie in b3.

---

[1] If pj were small, then the number of sub-simulation samples used to estimate vj would be correspondingly small. Thus, the more significant vj were in calculating the RS, the more accurately it would be estimated. This is a possible benefit of re-using the previously stored $\boldsymbol{Y}$ sample and partitioning it according to (also previously stored) values of the IAO rv as a means of calculating the RS of an IAO rv.

One reasonable way to choose the value of *di* within band bi is to require that, within the band, Prob(x > *di*) = Prob(x <= *di*). Thus *d1* = c^{-1} (*p1*/2), *d2* = c^{-1} (*p1* + *p2* / 2) and *d3* = c^{-1} (*p1* + *p2* + *p3* / 2), where c^{-1} is the inverse cumulative density function. So, we can calculate the *dj* from the inverse CDF (if explicitly known) or by calculating percentiles (if the CDF is not explicitly known).

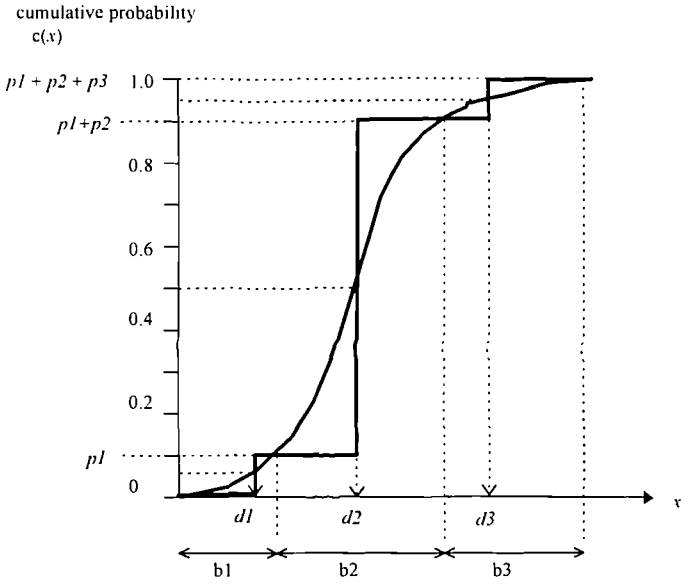


Figure 9-5: Approximating a continuous rv by a discrete one

Then *si* can be approximated by:

$$si = \text{var}(\boldsymbol{Y}) - (p1 \cdot v1 + p2 \cdot v2 + p3 \cdot v3)$$

Another reasonable way to choose the value of *di* within band bi, would be to require that the expected value of the discrete PDF should match the expected value of the original PDF, within each band. In this case, if the PDF is known explicitly, the *di* are given by:

$$pi \cdot di = \int_{x \in bi} x \cdot f(x) dx$$

and if they must be calculated from a sample, then *pi· di* can be approximated by the sample mean within band bi.

Clearly, taking this approximation to its extreme, and replacing ***xi*** with a single value *d0* with probability 1, we would have to choose *d0* = E(***xi***). Thus "the variance of ***Y*** | E(***xi***)" is an approximation to "the expected value of the variance of ***Y*** | ***xi***". However, if the {***xi***} are not independent, or if the distribution of ***xi*** is multi-modal, then this approximation may be highly inaccurate.

Both the methods of approximating a continuous random variable by a discrete one which are given above (using percentiles and equating expected values) involve choosing the probabilities first and then using the PDF (or sampled values) to calculate the corresponding discrete values. However, there may be situations

where it is preferable to choose the discrete values first, and then use the PDF or samples to calculate the probabilities. In particular, if a distribution has more than one peak and if the peaks are separated by zero-probability intervals, then it is important that the discrete values chosen should not lie within the zero-probability intervals. It may also be important that at least one discrete value is chosen from within each peak. Choosing discrete values which lie in the zero-probability intervals can be avoided by using fixed percentiles as described in the first method above (provided that interpolation between sample values is avoided when calculating the percentiles). An alternative method, which also avoids the zero-probability intervals, is to explicitly choose one value from each peak (or some other fixed number) and then calculate the corresponding probabilities.

9.2.1.4 Improvement 2: detect and use existing linearity

If it is known that the relationship between Y and xi is linear:

$$Y = a \cdot xi + b + F(x1,..x(i-1), x(i+1),...,xn)$$

where a and b are constants, F is some function of variables other than xi.

and that xi is independent of $x1$,.. then si is simply given by:

$$si = \text{var}(Y) - a^2 \text{var}(xi)$$

This simple case is likely to occur frequently (e.g. in cost aggregation). One possible approach would be to test for such a relationship by replacing all rvs in the model with their expected values and then evaluating Y with a small number of carefully chosen values of xi (e.g. 20th, 40th,...,80th percentiles). If the relationship is highly linear (as measured with a least squares regression) then we have already calculated a in our least squares regression, and can use it to give si. There is however a difficulty with this suggestion. Care must be taken over numeric random variables which are only used for particular IAO selections - if the appropriate IAO selection s happen to be the expected values, then the relationship will appear to be linear when it is not. And in general it is possible that the relationship appears highly linear, but becomes non-linear (or has a different linear coefficient) when some other variable in the model takes a value which is different from its expected value - i.e. interaction effects exist. For example, if $Y = a \cdot xi \cdot xj + b$, where the distribution of xj is discrete and is such that xj takes values which are far from its expected value. Note also that we have not yet considered the difficulties which arise when the xi are not independent of each other. Better approaches to regression are given in Section 9.2.2.3.

9.2.1.5 Fixing One Input at a Time

It is worth addressing at this point what the difficulties are with the intuitively obvious approach, mentioned at the end of "Improvement 1" above, of fixing one variable at a time at its expected value. This is quite computationally tractable, compared to the other direct calculation methods mentioned above. The algorithm to calculate si is therefore to perform n Monte Carlo simulations, each time holding one of the inputs at a fixed value, and record the variance in output for each simulation. An initial simulation is performed in which all of the inputs are varied. Then si, the risk sensitivity of the i'th input, is given by the reduction in

output variance achieved when ***xi*** was held at its fixed value. The difficulty here is deciding which value to fix the inputs at; the only reasonable choice is to use the expected value of the rv, but for IAO rvs in particular, simply choosing the alternative with the highest probability can yield misleading results. A second difficulty is that situations exist where fixing one of the inputs actually increases the output variance. An example of both of these difficulties is given in Appendix F. A third difficulty is that the expected value of ***xi*** may actually be a value which is never achieved (for a multi-modal distribution), and if there is also a highly non-linear relationship between ***xi*** and ***Y*** (such as a rule based method) then this will also give totally erroneous results.

9.2.1.6 Varying One Input at a Time

An even less computationally expensive, but very intuitive approach to obtaining the risk sensitivities is the algorithm shown below in pseudo-code where the n inputs are varied one at a time:

```
FOR k = 1 TO n
        hold all the xi at their expected values, except for xk
        perform a simulation in which only xk is varied (according to its PDF)
        sk = var(Y) - the variance in the resultant sample of Y
NEXT k
```

The major difficulty with this very simple algorithm, even for the case we are currently considering of independent $\{xi\}$, is that it ignores interactions between the $\{xi\}$ in the model: it may be, for example, that a very large variance in ***Y*** is only obtained when ***xk*** is varied and ***xj*** takes a particular value (which does not happen to be the expected value of ***xj***). This would not be detected by this simple algorithm. Even with independent random variables for the $\{xi\}$, such a circumstance could arise if F was highly non-linear, for example a rule-based method of the form:

```
if (xj ∈ [a, b]) AND (xk ∈ [c, d])
      result = 1000 * xk * xk
else
      result = 0
```

where the expected value for ***xj*** lies (just) outside the interval [a, b] and the expected value for ***xk*** lies (just) outside [c, d]. The risk sensitivities calculated for ***xk*** and ***xj*** would be 0, which is not correct.

9.2.1.7 Quantifying Interaction Effects

This raises the question: should risk sensitivities be calculated for each *combination* of inputs, rather than solely for each individual input - in other words, should the interactions (the cross-terms in a linear model) be quantified? This would add considerably to the computational complexity, but it would not necessarily make the results too difficult to interpret. Highlighting cases where, for a example, a pair of rvs were jointly responsible for much of the variance in the output, but neither were individually major contributors would certainly be useful. Consider the example, referred to earlier, given in Appendix F. The ideal result from a risk sensitivity analysis of such an example would be that the joint risk sensitivity to the pair of RVs was larger than either of their individual sensitivities. In this thesis, however, in the interests of simplicity, only

risk sensitivity to individual inputs is considered and the definition of RS given at the beginning of this section is adhered to.

A more specific formulation of the problem, making use of the object structure, is proposed in Section 9.2.3 below. However, before examining this, we briefly review the basic techniques available for risk sensitivity analysis using the simple problem formulation above. The aim of section 9.2.2 is to identify other techniques which are less computationally expensive than the direct calculation methods described above, but which may yield comparable values for the risk sensitivities.

9.2.2 Available Numerical Techniques

There are two categories of numerical techniques which can be used to perform sensitivity analysis where the relationship between the inputs and the output is not known in a closed form: What could be termed "dynamic" techniques involve driving the model inputs and thus require control of the model - examples include the kind of "direct calculation" techniques described in the previous section, statistical experimental design in general and also Fourier techniques. The other category could be termed "passive" techniques, in that they only rely on recording the model inputs and output, and thus can be used to analyse models where the analyst does not have the power to drive the inputs - for example when analysing social, political or medical systems - or where it is computationally desirable to re-use existing samples for the inputs and output, as in a large Monte Carlo simulation. The two techniques in this category are regression and correlation, which are very closely related. In analysing a risk model, we are able to use dynamic techniques but it may prove computationally more efficient to use passive ones if previously generated samples can be re-used.

9.2.2.1 Statistical Experimental Design

In statistical experimental design (see [Law 1991, page 656]) the inputs are termed "factors" and the output is termed the "response". A set of experiments are performed, each experiment using a different set of values for the factors, and the response is measured for each experiment. The experiments are designed so that statistics of interest can be calculated from the results. The direct calculation methods given in Section 9.2.1.2 are examples of how statistical experimental design can be used to calculate sensitivities. There are however far more efficient algorithms available, and the simplest of these which illustrates the concept is "**2^n factorial design**".

Here, only two levels are chosen for each of the factors (usually denoted by '+' and '-'), and the response is obtained for each possible combination of factor levels; thus if there are n factors, the response is evaluated 2^n times. The levels chosen for xi could for example be $\mu i + \sigma i$ and $\mu i - \sigma i$, where μi is the mean value of the PDF for xi and σi is its standard deviation. The set of factor levels for an experiment is termed a design point. The sensitivity sk (termed the *main effect* of the k'th factor) is then given by the average change in the response due to the kth factor (xk). For example, suppose n = 3 and the experimental results obtained were as shown in Table 9-1.

Experiment number	Factor 1 (*x1*)	Factor 2 (*x2*)	Factor 3 (*x3*)	Response (*Y*)
1	-	-	-	*y1*
2	+	-	-	*y2*
3	-	+	-	*y3*
4	+	+	-	*y4*
5	-	-	+	*y5*
6	+	-	+	*y6*
7	-	+	+	*y7*
8	+	+	+	*y8*

Table 9-1: 2^n Factorial Experimental Design

Then the main effect of ***x1*** is given by:

Equation 9-1

$$s1 = \frac{(y2 - y1) + (y4 - y3) + (y6 - y5) + (y8 - y7)}{4}$$

This can be regarded as the average change due to ***x1*** or, alternatively, as the difference between the mean when ***x1*** = + and the mean when ***x1*** = -. If we suppose that the two values selected for a factor (+ and -) represent an approximation of the continuous random variable to a discrete rv with two equi-probable values of + and -, then how does the value of *s1* given above relate to the "expected value of the reduction in variance" which is approximated in the direct calculation methods of Section 9.2.1.2? In this case, the expected variance is given by:

$$\begin{aligned} &E(var(y|\mathbf{x1})) \\ &= Pr(x = +) \times var(y|x = +) + Pr(x = -) \times var(y\ x = -) \\ &= \frac{1}{2} \times \frac{1}{4-1} \sum_{i=2,4,6,8} (yi - \mu_{x+})^2 + \frac{1}{2} \times \frac{1}{4-1} \sum_{i=1,3,5,7} (yi - \mu_{x-})^2 \end{aligned}$$

And thus, using the definition of RS given in Section 9.2.1.1, we obtain:

$$s1 = var(y) - E(var(y|\mathbf{x1})) = \frac{1}{n-1} \sum_{i=1}^{n} (yi - \mu)^2 + \frac{1}{6} \left[\sum_{i=2,4,6,8} (yi - \mu_{x+})^2 + \sum_{i=1,3,5,7} (yi - \mu_{x-})^2 \right]$$

where

$$\mu = \frac{1}{n} \sum_{i=1}^{n} yi$$

and

$$\mu_{x+} = \frac{y2 + y4 + y6 + y8}{4}$$

and

$$\mu_{x-} = \frac{y1 + y3 + y5 + y7}{4}$$

It is clear that the *main effect* of ***x1*** and the *expected variance reduction* are quite different measures:. In particular:

1. The algorithms are different. The main effect algorithm can be summarised as:

- fix all ***xi*** except ***x1***
- vary ***x1*** and measure the resultant variation in ***Y***
- repeat for a representative sample of values of all variables
- take expected value (mean) of variations in ***Y*** obtained

Whereas most of the algorithms given Section 9.2.1 to calculate expected variance can be summarised as:

- fix ***x1***
- vary all other ***xi*** except ***x1*** and measure resultant variation in ***Y***
- repeat for a representative sample of values for ***x1***
- take expected value of variations in ***Y*** thus obtained
- subtract from overall variation in ***Y***

The exception being the "varying one input at a time" algorithm in Section 9.2.1.6 which is comparable to the main effect algorithm.

2. The way in which the variation of ***Y*** is measured is different. For the main effect, the variation is defined as the range of values obtained - this is reasonable when the variation is only measured over two values as above, but would not generalise well if more than two values were used. For the expected variance, the variation is defined as the statistical variance, which can be used for any reasonably large number of ***Y*** values. If one were to use the main effect to calculate RS, one would certainly need to take the absolute value of the range, rather than the (signed) definition given in Equation 9-1. Otherwise, factors which had a strong positive correlation with ***Y*** in some circumstances and a strong negative correlation under others, would have their positive effects cancelled out by their negative ones and give a small response; despite always making a large contribution to the variation in ***Y***.

What are termed two-factor *interaction effects* can also easily be computed from the same set of experimental results: the size of the interaction effect between factors j and k provides a measure of the extent to which the effect of factor j depends upon factor k. Higher factor interaction effects can also be computed, up to k-factor.

In this kind of experimental design, the experimental factors do not always necessarily include all possible sources of uncertainty and thus some of the variance in $\boldsymbol{Y}$ obtained may not in fact be due to $\{\boldsymbol{xi}\}$, but to other factors. This difficulty is usually overcome by replicating the experiments several times at each design point (i.e. at each set of factor values) and calculating confidence intervals for the responses obtained. In calculating risk sensitivity however, we wish to include all possible sources of uncertainty and this is not an issue.

This kind of experimental design could only be directly applied to numeric-valued rvs, not to IAO rvs. The advantages of a 2^k factorial design, for example, over the type of Monte Carlo direct calculation methods given in Section 9.2.1 are twofold. Firstly, a single "sub-simulation sample" (or set of response values) is used to evaluate all the *sk*, whereas the direct calculation methods perform a different sub-simulation for each *sk*. Secondly, it can be guaranteed that when calculating the effect of variations in $\boldsymbol{xk}$ on $\boldsymbol{Y}$, all combinations of the chosen values (for example $\mu i + \sigma i$ and $\mu i - \sigma i$) for the other $\boldsymbol{xi}$ will be used to evaluate si. In the direct calculation methods we rely on Monte Carlo random sampling from within the continuous range of $\boldsymbol{xi}$ values to generate as many combinations as possible. Thus a 2^k factorial design is better where there are significant interactions in the model. The simple designs in Section 9.2.1 would however be better suited to a situation where the shape of the probability distributions of the inputs is highly significant. The direct calculation method would also be better in the case where the model is highly non-linear and non-monotonic, since in this case experiments performed at intermediate levels become necessary to maximise the change in the response. The 2^n factorial design would be more efficient for small values of n, but becomes prohibitively expensive for large n.

One approach generally used in statistical experimental design to reduce the computational burden where there are many factors is to perform what are termed *fractional* factorial designs. Here, only 2^{k-p} experiments are performed instead of the full 2^k experiments. There is much literature devoted to how one should choose the experiments - but all such experimental designs basically rely on assuming that some of the interaction effects are small compared with the main effects (see [Law 1991, p. 670 onwards]). In a fractional design, what is termed *confounding* occurs. This means that two or more effects are inseparable because they are both (or all) given by exactly the same algebraic expression. If, for example, a three-way interaction is confounded with a main effect, then one wishes to assume that the three way interaction is small compared to the main effect. The idea is to choose the experiments so that the assumptions are reasonable. A fractional design has what is termed a *resolution*. If a fractional design has a resolution of R then two effects will not be confounded if the sum of their "ways" is less than R (where a main effect is 1-way, etc.). Once one has chosen k, p and R, it is possible to construct a suitable fractional experimental design using a set of rules ([Law 1991, page 672] for example).

Another approach, used when the number of factors (k) is larger than the number of experiments which can be performed (E), is to choose the values for each factor independently and randomly, subject only to the following condition: that, over the set of all the experiments, each factor should take an equal number of + and - values. In other words, if a table similar to Table 9-1 were to be drawn up for the experiments, each column would contain $E / 2$ values of + and $E / 2$ values of -. For certain values of k and E this approach can be improved by choosing the factor values, not at random, but so as to minimise the confounding which occurs (see [Mauro 1986]). The experiments involved in evaluating a risk model are not, however, sufficiently expensive so as to justify such "supersaturated" experimental designs.

9.2.2.2 Fourier methods

The Fourier technique proposed in [Schruben 1986] requires that the random variable inputs to the model are varied sinusoidally, with each input at a unique frequency, and the frequency spectrum of the output is analysed using a Fourier transform (numerically achieved using the Fast Fourier Transform algorithm). However, if the model is non-monotonic and discontinuous then the technique will fail because in this case a peak in the frequency spectrum of the output at a frequency ω cannot necessarily be attributed to that input which was varied at frequency ω. Another disadvantage of this technique is that the shape of the probability distribution of the input variables cannot be taken into account - only their overall variance.

9.2.2.3 Regression and Correlation

Using regression, a function G of known closed form (usually linear) is fitted to a sample of $\boldsymbol{Y}$ values as a function of the $\boldsymbol{xi}$ (a planar equation) and the variance in $\boldsymbol{Y}$ due to $\boldsymbol{xk}$ can then be estimated by:

$$\left(\frac{\partial\ G(\boldsymbol{x1},...,\boldsymbol{xn})}{\partial\ \boldsymbol{xk}}\right)^2 * \sigma^2(\boldsymbol{xk})$$

where $\sigma^2(\boldsymbol{xk})$ is the variance of $\boldsymbol{xk}$. Unlike the experimental design and Fourier methods, this passive technique could use the same stored samples of $\boldsymbol{Y}$ and $\{\boldsymbol{xi}\}$ which were generated during a simulation. However, it would probably only be computationally feasible if G were assumed to be linear. Regression would give misleading results if the relationship between $\boldsymbol{Y}$ and $\{\boldsymbol{xi}\}$ were discontinuous. The method of least squares is the most conceptually simple method to obtain the coefficients for G. This involves choosing those coefficients which will minimise the value S, where:

$$S = \sum_{J} \left(G(x1_j,...,xn_j) - Y_j\right)^2$$

So S is a measure of goodness of fit - the mean squared distance of the sampled values from the best fit surface, measured in the $\boldsymbol{Y}$ direction. If G is a linear function:

$$G(\boldsymbol{x1},...,\boldsymbol{xn}) = bo + \sum_{i=1}^{n} bi \cdot \boldsymbol{xi}$$

then it can be shown ([Hayes 1970, p.69]) that the vector $\mathbf{b} = [b0,..,bn]^T$ is given by:

Equation 9-2

$$\mathbf{b} = \left(\mathbf{X}^t\mathbf{X}\right)^{-1}\mathbf{X}^t\mathbf{Y}$$

where the matrix **X** contains the sampled *xi* values and the vector **Y** contains the sampled *Y* values:

$$\mathbf{X} = \begin{bmatrix} 1 & x1_1 & \dots & xn_1 \\ 1 & x1_2 & \dots & xn_2 \\ \dots & \dots & \dots & \dots \\ 1 & x1_M & \dots & xn_M \end{bmatrix}$$

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ \dots \\ y_M \end{bmatrix}$$

The variance due to *xi* (risk sensitivity to *xi*) can then be estimated by $bi^2 \cdot \sigma_{xi}^2$. *Step-wise* linear regression is a computational method for calculating **b** which adds or deletes input variables to the analysis one at a time, discounting the "less significant" variables - i.e. those which have the smallest effect on the variance of the output.

A closely related technique is to use the estimated correlation coefficient between ***Y*** and ***xk*** (estimated from samples) as a measure of the risk sensitivity of ***Y*** to ***xk***. In fact, using the linear correlation coefficient to calculate variance reduction (as described below) is exactly the same as performing a linear regression of each ***xk*** against ***Y*** separately. The linear correlation coefficient takes a value between -1 and 1 and it provides a measure of the strength of a linear relationship between a pair of random variables. It takes a value of 1 if the values taken by the variables lie on a perfect straight line with positive slope and a value of -1 if they lie on a perfect straight line with negative slope. A value close to 0 indicates that the rvs are independent. The linear correlation coefficient is defined as:

$$\rho_{Yxk} = \frac{cov(\boldsymbol{Y},\boldsymbol{xk})}{\sigma_Y \sigma_{xk}}$$

where cov(***X***,***Y***), the covariance of ***X*** and ***Y***, is defined by:

$$cov(\boldsymbol{X},\boldsymbol{Y}) = E[(\boldsymbol{X} - \mu_X)(\boldsymbol{Y} - \mu_Y)] = E(\boldsymbol{XY}) - E(\boldsymbol{X})E(\boldsymbol{Y})$$

The sample correlation coefficient, also known as Pearson's r, is an estimator for the linear correlation coefficient. It is defined as:

$$r_{XY} = \frac{\sum_i (Xi - Mx)(Yi - My)}{N \cdot Sx \cdot Sy}$$

where Mx is the sample mean value of $\boldsymbol{X}$, Sx is the sample standard deviation of $\boldsymbol{X}$ and similarly for $\boldsymbol{Y}$. It can be shown ([Hayes 1970, pp. 23-25] for example) that ρ_{XY} is very closely related to the linear regression curve between $\boldsymbol{X}$ and $\boldsymbol{Y}$. When the regression curve is linear, then:

$$E(\boldsymbol{Y} \mid x) = \mathrm{a} + \mathrm{b} \cdot x$$

where a and b are constants. In this case a and b are given by:

Equation 9-3

$$\mathrm{b} = \rho_{XY} \cdot \frac{\sigma_Y}{\sigma_X}$$

$$\mathrm{a} = \mu_Y - \rho_{XY} \frac{\sigma_Y}{\sigma_X} \mu_X$$

and so the variance in $\boldsymbol{Y}$ which is due to variance in $\boldsymbol{X}$ is $\mathrm{b}^2 \cdot \sigma_X^2$ which, by Equation 9-3, is equal to $\rho_{XY}^2 \, \sigma_Y^2$ and the remainder, which is not accounted for by the linear regression is given by:

$$var(\boldsymbol{Y}|x) = \sigma_Y{}^2 \cdot \left(1 - \rho_{XY}{}^2\right)$$

In other words, the square of the linear correlation coefficient relating $\boldsymbol{X}$ and $\boldsymbol{Y}$ is the proportion of the variance in $\boldsymbol{Y}$ which can be accounted for by a linear regression against $\boldsymbol{X}$. Therefore, if we could assume that $\boldsymbol{Y}$ was linearly related to the $\{\boldsymbol{xi}\}$ then the linear correlation coefficient could be used to calculate the risk sensitivity as defined in Section 9.2.1.1 (as the expected value of the variance reduction).

Unsurprisingly, if we compare the algebraic expression for the coefficients of the least-squares best fit straight line through a sample with Equation 9-3 we find a marked similarity. If a' and b' are the coefficients of the best fit straight line through a sample $\{Xi, Yi\}$ then it can easily be shown that:

$$b' = r_{XY} \cdot \frac{S_Y}{S_X}$$

and

$$a' = M_Y - r_{XY} \frac{S_Y}{S_X} M_X$$

In situations where this linearity assumption cannot be made, the *rank correlation coefficient* (also known as the Spearman Rank-Order Coefficient or Rs) is generally utilised. Like the linear correlation coefficient, the rank correlation coefficient takes a value between -1 and + 1 but it provides a measure of the strength of a relationship between two random variables even if the relationship is not linear. If the values of two rvs are highly correlated but lie on a curve rather than a straight line, then the linear correlation will be small but the rank correlation will be large. The rank correlation coefficient for a sample is defined as follows. Each value of each variable in the sample is assigned a "rank", representing its position in an ordering of the sample according to size: the largest X value will be assigned a rank of 1, the second largest, a rank of 2, and so on. Similarly for the Y sample values. Then the (linear) correlation between these rank values is calculated, rather than the correlation between the sample values themselves. Although this provides a measure of the strength of a non-linear relationship (provided it is monotonic), it unfortunately does not provide any measure of $\mathrm{Var}(Y \mid x)$. It is obvious that rank correlation cannot help us in predicting $\mathrm{Var}(Y \mid x)$, since it tells us nothing of the shape of the regression curve. Thus, although rank correlation provides a useful measure of the strength of a monotonic relationship, it is not commensurable with risk sensitivity as defined in Section 9.2.1.1.

Returning to the linear correlation coefficient, the discussion so far has centred on a bi-variate distribution. In analysing the risk sensitivities of the attributes in a risk model, we have a multi-variate distribution. If (as we are assuming in this Section) the $\{xi\}$ are independent of one-another, there is no difficulty in using ρ_{Yxi} as a measure of the variance reduction which would be achieved by fixing xi (apart from the linearity assumption itself). However, if there may be dependencies between the $\{xi\}$, then this is no longer strictly true. Suppose $x1$ is strongly linearly related to $x3$. The size of the correlation between $x1$ and Y will no longer indicate the variance reduction to be expected from fixing $x1$ - the influence of $x3$ on that correlation must be removed. In this case, the *partial correlation coefficient* should be used. This is discussed in Section 9.2.3, where the independence of the inputs is no longer assumed.

An important if obvious point about the above regression and correlation techniques is that they rely on the concept of size - thus they are only applicable to numerical rvs, not to IAO rvs. The value domain of an IAO rv is effectively an arbitrary object identifier - thus a sample taken from an IAO rv provides only information about the category, not the size, of the rv (this is termed categorical data). If one were to calculate the linear (or indeed the rank) correlation coefficient relating an IAO rv to the output, the results obtained would depend entirely upon this arbitrary ordering of objects. Suppose an IAO models a choice between objects A, B and C where A costs more than B costs more than C, and the output attribute is the sum of their costs. If the object identifiers assigned to {A, B, C} are {0, 1, 2} then a -ve correlation will be obtained, if they are {2, 1, 0} then a +ve correlation will be obtained and if they are {1, 2, 0} then a correlation close to zero will be obtained.

Techniques do however exist for drawing statistical inferences from categorical data (see [Hayes 1970, Chapter 12] for example). Statistical tests for independence between categorical data are described which test the hypothesis that two categorical variables are independent, using the χ^2 test. However, in our case only one of the variables is categorical and the other (the output) is numerical, and we also require a measure of the strength of association, not merely its existence. Measures of statistical strength of association between

pairs of categorical variables exist (for example the index of predictive association, see [Hayes 1970, p. 211]), but in our case we wish to measure the strength of association between a categorical variable and a numerical one. Therefore, for categorical rvs, a "direct calculation" method (Section 9.2.1.2) must be used.

9.2.2.4 Conclusions Concerning Basic Numerical Techniques with Independent Inputs

There appears to be no single technique or algorithm which is computationally feasible and which will provide the correct risk sensitivity values according to the definition given in Section 9.2.1.1 under all circumstances.

Direct calculation methods can be applied to both categorical (IAO) rvs and numerical rvs and are most applicable where:

- distribution shape is important (e.g. multi-modal distributions)
- there are few significant interactions (relies on random sampling to detect interaction effects)
- there is a non-linear relationship between the inputs and output

2^n factorial statistical experimental design methods can only be applied directly to numerical rvs and are most applicable where:

- distribution shape is unimportant (distributions are single-peaked)
- there are significant interactions and we wish to quantify them separately
- the relationship between the inputs and output is monotonic and close to linear
- the number of inputs is not very large

Linear correlation can only be applied to numerical rvs and is most applicable where:

- the relationship between the inputs and output is linear
- re-use of existing samples is required (for computational efficiency or because we can't drive the model)

Rank correlation can only be applied to numerical rvs and is most applicable where:

- the relationship between inputs and output may not be linear
- re-use of existing samples is required

Fourier techniques can only be applied to numerical rvs and are most applicable where:

- there is a linear relationship between the inputs and the output
- the input distribution shape is unimportant (distributions are single-peaked)
- the number of inputs is very large and we only wish to identify the most significant ones

All the basic techniques above require that the inputs are independent of each other, although partial correlation coefficients can be used when this condition is not met.

Direct calculation and linear correlation methods provide commensurable measures of risk sensitivity, rank correlation provides a different measure and 2^n factorial statistical experimental design methods provide a third set of measures (main effects and interactions). The main effect in a 2^n factorial experiment is commensurable with the results obtained using Fourier techniques. Each measure conveys information which is of relevance under certain circumstances when identifying the major sources of uncertainty in a risk model.

9.2.3 Problem Formulation Using Object Structure

If we formulate the problem as described in Section 9.2.1 (without considering the object structure) then the function F will generally be highly discontinuous and non-monotonic because of the effect of the IAO rvs, but the inputs will be independent. This limits the applicable numerical techniques to rank correlation. The amount of memory required to store samples for every rv in the model may well be prohibitive.

The other difficulty with formulating the problem in this way is that a large model will contain several hundred random variables (at least) and a ranking of every single random variable is really too detailed to provide the user with a useful overview of the risk content of the whole project. It may be the case that the individual rv which makes the largest contribution to the uncertainty in overall vehicle piece cost is, for example, the cost of a particular engine component. However, the accumulated effect of many rvs in the electrical area may mean that the electrical area makes a much larger contribution to the overall risk than the engine area. The user who carries out a risk sensitivity analysis of the entire vehicle will generally be more concerned with the relative contributions from each area than from each individual component.

Thus, it is proposed that a containment hierarchy should be used to decompose the risk sensitivity analysis in some way. Selected objects in the risk model will be marked as "containers" and these containers will form a hierarchy - an obvious choice for the containers is the nodes in a "bill-of-materials" view of the risk model, i.e. the major assemblies and sub-assemblies. Each risk sensitivity analysis performed will only apply to a particular container - the contribution from an attribute belonging to a child in the containment hierarchy will be regarded as a single source of uncertainty.

In certain very simple cases, the containment hierarchy structure will correspond directly to the attribute derivation network, and in this situation a numerical evaluation of risk sensitivity may not be necessary and a simpler approach involving visual layered cumulative distribution functions may be adopted instead.

In the simplest case there may be a single interface attribute (cost for example) associated with each container, and the relationship between the cost of a container (Y) and that of its children ($x1$, $x2$,..) may be known to be of the form $Y = f1(x1) + f2(x2) + ..+ f3(xn)$. In this case, the cumulative probability distribution for Y may be plotted with the effect of including each of the children's costs being added in one at a time, as shown in Figure 9-6, providing a visual indication of the contribution to overall cost uncertainty made by each child.

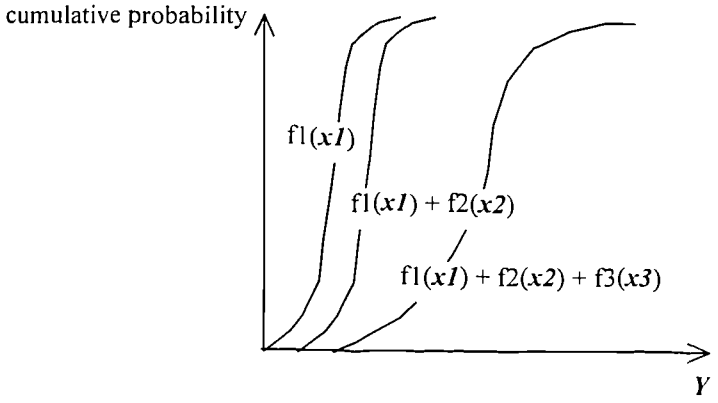


Figure 9-6: Visual approach to risk sensitivity using layered CDFs

More generally, however, the relationship will not be of the simple form shown above and the situation may be further complicated by the presence of IAO (categorical) random variables. Thus a more general approach is required.

In the general case, the impact which the proposed use of a containment hierarchy has on the problem formulation compared to that presented in Section 9.2.1 is firstly to reduce the number of random variables involved but secondly to invalidate the assumption that the inputs are independent of each other. Some of the input rvs (those which are either used by other containers or which belong to other containers, which are termed the *interface attributes*) no longer have a known form for their pdf and parameters - but sample values can be obtained for these inputs. And the interface attributes are not necessarily uncorrelated with each other.

It is proposed that during an initial simulation of the entire model, the randomly generated samples for the interface attributes are stored to disk. If the model is subsequently edited by the user, only those interface attributes which are affected by the edit are re-simulated and re-stored. This proposal reduces the computational cost of minor changes to the risk model and also means that pre-computed sample values are available for all interface attributes during a risk sensitivity analysis of a particular container.

The problem can thus be re-formulated:

$$Y = F(x1,...,xn)$$

where each input random variable xi may be either:

- a *local numerical rv*: This could either be defined directly by the user or created by a heuristic method, but in both cases the PDF and its parameters are known and no pre-computed sample is stored. It is only used within this container object.
- or a *local categorical (IAO) rv*: This is a discrete integer-valued random variable used to choose between alternatives in a Random IAO and is only used within this container.

- or a *local numerical interface attribute*: This is an attribute which "belongs" to this container, but is used by other container objects. A pre-computed sample is available, and it may or may not also have a known PDF and parameters.
- or a *remote numerical interface attribute*: This is an attribute which "belongs" to another container, but is used inside this container. A pre-computed sample is available, and it will not have a known PDF and parameters.

Correlations may exist between the interface attributes, but the non-interface attributes are independent of each other and of the interface attributes. By following the attribute dependency tree it is possible to determine where correlations may exist between interface attributes, and where interface attributes are guaranteed to be independent.

There is a further complication in the situation where *This* container uses attributes of a set of other containers which are alternatives to each other and where the IAO/s do not belong to *This*. (Note that there may be more than one nested IAO). In order to carry out the RSA, it is clear that *This* will need access to such IAOs. The IAO rvs can be regarded as interface attributes, in that their samples should be stored during an ordinary simulation. Thus we must add a fifth category of input rv:

- a *remote categorical rv:* This is an IAO rv which is used by this container but belongs to another container. It will have a known PDF and parameters and a pre-computed sample will also be available.

The impact which the above mentioned changes in the problem formulation (availability of pre-computed samples for inputs and existence of correlations between inputs) have on the numerical techniques which are applicable is investigated in the two sections below.

9.2.3.1 Impact of Decomposing RSA: Samples for Inputs

In this section we temporarily place on one side the difficulties caused by the existence of correlations between the inputs, and consider the numerical techniques which are applicable when some of the random variable inputs to the container model are represented by PDFs and their parameters, but others are represented only by pre-computed samples, stored during previous simulations of the model.

The "direct calculation" method for numerical rvs given in Section 9.2.1.3 involves approximating a continuous PDF by a discrete one. When the PDF itself is not known, but a pre-computed sample is available, and the underlying PDF has multiple peaks separated by zero-probability intervals, this involves particular difficulties. It was suggested in Section 9.2.1.3 that one value should be chosen from within each peak and then the sample should be used to calculate the probability corresponding to each discrete value. When the PDF is known in closed form (e.g. as a piece-wise linear function) this is straightforward and may well be a good approach.

When the PDF is not known, and we have only a pre-computed sample for a remote numerical *interface* attribute, we would ideally like to adopt a similar strategy and choose at least one value from each peak when approximating the continuous rv by a discrete one. Thus the sample must be divided into peaks each of

which contains no zero-probability intervals. One possible method for example is to sort the sample into ascending order and look for gaps between consecutive values which exceed, for example, 10% of the entire sample range. However, there is obviously a danger that this method would be very inefficient where the underlying distribution is actually uni-modal but long-tailed - dividing the tail into several very low probability discrete peaks. It is not clear how an algorithm could be designed which successfully identifies the zero-probability intervals from a multi-modal sample, and it is reluctantly concluded that it cannot be guaranteed that one discrete value will be chosen from within each peak. Instead, fixed percentiles of the sample must be used (avoiding interpolation as noted in Section 9.2.1.3). Thus there may be alternatives in other containers which are not explored directly during a risk sensitivity analysis.

9.2.3.2 Impact of Decomposing RSA: Correlated Inputs

If there are correlations between the risk model inputs, then the dynamic numerical techniques discussed in Section 9.2.2 are no longer applicable. Fourier techniques, "direct calculation methods" involving sub-simulations and statistical experimental design would all be impossible where such correlations exist because they all rely on driving one input at a time.

Suppose inputs $x1$ and $x2$ are correlated. If the linear correlation technique is to be used to quantify the risk sensitivity of Y to each of them then it is no longer sufficient to calculate the linear correlation coefficients ρ_{Yx1} and ρ_{Yx2}. Because the input variables are correlated, a *partial correlation coefficient* (see [Hayes 1970, pp.65 - 67] and [Rao 1973, pp.268 - 270]) must now be calculated. Partial correlation coefficients can be used to represent the degree of association between Y and $x1$ but with the influence of $x2$ removed - as though $x2$ had been held constant. The partial correlation coefficient $\rho_{Yx1.x2}$ is the correlation between Y and $x1$, adjusted to remove the effect of $x2$. It is generally defined as the correlation between e_Y and e_{x1} where e_Y is the residual after subtracting the regression of Y against $x2$ from the sampled Y values, and e_{x1} is the residual after subtracting the regression of $x1$ against $x2$ from the sampled $x1$ values:

$$e_Y = Y - (b0 + b1 \cdot x2)$$
$$e_{x1} = x1 - (b0' + b1' \cdot x2)$$

Or, where the influence of more than one variable should be removed:

$$e_Y = Y - (b0 + b1 \cdot x2 + \ldots + bn \cdot xn)$$
$$e_{x1} = x1 - (b0' + b1' \cdot x2 + \cdots + bn' \cdot xn)$$

The partial correlation coefficient $\rho_{Yxi.x1\ldots xn}$ represents the proportion of the variance in Y which is attributable to the linear regression of Y on xi with the other $\{xj\}$ held constant. The partial correlation coefficients are closely related to the multiple linear regression solution - it can be shown that bi, the coefficient of xi in the multi-linear regression equation for Y, is given by:

$$bi = \rho_{Yxi.x1...xn}\left(\frac{\sigma_{Y.x1...xn}}{\sigma_{xi.x1...xn}}\right)$$

(Note that this is analogous to Equation 9-3.) If *by* is the coefficient of ***Y*** in the multi-linear regression equation for ***xi*** then it follows from the above equation, by symmetry, that:

$$\rho_{Yxi.x1...xn}{}^2 = bi \cdot by$$

and thus the partial correlation coefficient can be calculated from the results of two multi-linear regressions. This is the technique which has been initially implemented in the risk tool.

The partial correlation coefficients can also be calculated from the linear correlation coefficients between each pair of variables (sometimes termed the zero order correlation coefficients), and this is the usual means of computing them by hand. It can be shown that:

$$\rho_{Yx1.x2} = \frac{\rho_{Yx1} - \rho_{Yx2}\rho_{x1x2}}{\sqrt{\left(1-\rho_{Yx2}{}^2\right)\left(1-\rho_{x1x2}{}^2\right)}}$$

and similarly:

$$\rho_{Yx1.x2x3} = \frac{\rho_{Yx1.x3} - \rho_{Yx2.x3}\rho_{x1x2.x3}}{\sqrt{\left(1-\rho_{Yx2.x3}{}^2\right)\left(1-\rho_{x1x2.x3}{}^2\right)}} = \frac{\rho_{Yx1.x2} - \rho_{Yx3.x2}\rho_{x1x3.x2}}{\sqrt{\left(1-\rho_{x1x3.x2}{}^2\right)\left(1-\rho_{x1x3.x2}{}^2\right)}}$$

Thus, by repeatedly applying the same formula, any partial correlation coefficient can eventually be computed from the zero order coefficients. It is possible that this would prove a more computationally efficient algorithm in the future.

9.2.4 Proposed Algorithm

It is proposed for reasons given in Section 9.2.3, that a containment hierarchy should be defined and used to decompose both simulation of the entire model and the risk sensitivity analysis problem. The user will perform a risk sensitivity analysis at a particular level in the containment hierarchy. The analysis may identify an attribute of another container (a remote numerical interface attribute) as a major contributor. In this case, the user may then choose to perform a further risk sensitivity analysis on that attribute, and so on.

It has been argued in the previous sections that there is no single measure of risk sensitivity which is universally applicable, and that the best solution will be to provide several different, alternative RS methods to the user, each of which may yield a different ranking of the sources of uncertainty in the model. The solutions described in this section are only two of these methods.

Two solutions are proposed - neither of which attempts to quantify the interaction effects. The first solution is based on a combination of linear correlation and direct calculation (these being commensurable) and is applicable to all types of rv in the model. The second solution is based on rank correlation and will only be applied to the numerical rvs, not to categorical (IAO) rvs.

9.2.4.1 Direct Calculation Solution

It is proposed that those Sim object methods which are linear with respect to each individual input random variable should be explicitly identified as such by the programmer who builds the Sim classes. This is a safer (and computationally more efficient) approach than attempting to detect linearity at run-time and will be applicable to many of the methods used in a typical risk model. It is also proposed that the programmer should identify those methods which are severely non-linear and discontinuous: namely, the rule-based methods. Although no foolproof special algorithm can be applied in the latter case, the user can be warned that the results may be unreliable.

It is also proposed that the risk tool should automatically detect the dependencies between interface attributes and store this information in the risk model. When the user specifies an output attribute in the risk model, the risk tool should build and analyse the complete attribute dependency tree for that attribute, down as far as the independent random variables. The interface attributes should be identified and dependencies between such interface attributes should be noted. The risk tool should only store the existence of a dependency, not attempt to quantify it. When the whole model is simulated, the risk tool should store sample values for the interface attributes and for the output attribute/s.

When a risk sensitivity analysis is performed, the risk tool should use a different technique depending upon the nature of the input variable. Another complete simulation of this container is performed (with the same number of iterations as when the whole model was simulated) but the interface samples are re-used and only the independent leaf variables in this container are re-sampled (the local rvs). This is referred to below as "the first container sub-simulation". The independent leaf samples for case 1. (below) are stored (temporarily), as are the IAO samples along with the new $\boldsymbol{Y}$ sample. If there are any variables satisfying 3. below, then further container sub-simulations are performed for each of them. There are six possible cases for an input variable which are relevant to how the variance reduction for that variable is calculated:

1. if ***xi*** is a non-interface local numerical random variable (a leaf rv within this container) which is related to ***Y*** by a linear method:

> The variance reduction is given by the linear correlation coefficient between ***xi*** and ***Y***, which is calculated from the local leaf sample and the new ***Y*** sample.

2. else, if ***xi*** is a local/remote numerical interface rv which is related to ***Y*** by a linear method and which is independent of the other inputs:

> The variance reduction is given by the linear correlation coefficient between ***xi*** and ***Y***, which is calculated from the re-used interface sample and the old ***Y*** sample.

3. else, if ***xi*** is a local, non-interface numerical rv (but not known to be related linearly to ***Y***) or ***xi*** is a local interface numerical attribute which is independent of all other input variables and whose PDF is known:

> The continuous rv is approximated to a discrete one and the "direct calculation" method is used to calculate the variance reduction. The way in which the discrete rv is chosen will depend upon the particular distribution function of ***xi*** (in particular, if it is piece-wise linear and multi-modal then a discrete value in each peak should be used). The pre-computed samples for interface attributes will be re-used as will the case 1. and IAO samples stored in the first container sub-simulation.

4. else, if ***xi*** is a local/remote categorical random variable (an IAO rv):

> The new ***Y*** sample is partitioned according to the stored IAO values, and the variance reduction is calculated directly; the variance of each partitioned sub-set of the ***Y*** sample is calculated and the expected value of the variance is then given by weighting the variances with their probabilities (given in ***xi***).

5. else, if ***xi*** is a numerical interface attribute which is independent of all other input variables and whose PDF is not known:

> ***xi*** is approximated by a discrete rv at fixed percentiles of the sample (e.g. 10%, 50% and 90%) and the expected variance reduction calculated by direct calculation.

6. else, if ***xi*** is a numerical interface attribute which is known to be correlated with one or more other input variables:

> The partial correlation coefficient is used, discounting the effect of those attributes with which ***xi*** is known to be correlated.

The cases identified above are illustrated in Figure 9-7 which shows that they are exhaustive. All possible categories for the input attributes are listed using a taxonomy tree, and then for each category (each leaf of

the taxonomy tree, shown bolded) one of the above cases is identified (the case number is shown beneath the bolded leaf node). Table 9-2, below, contains a summary of the cases identified above.

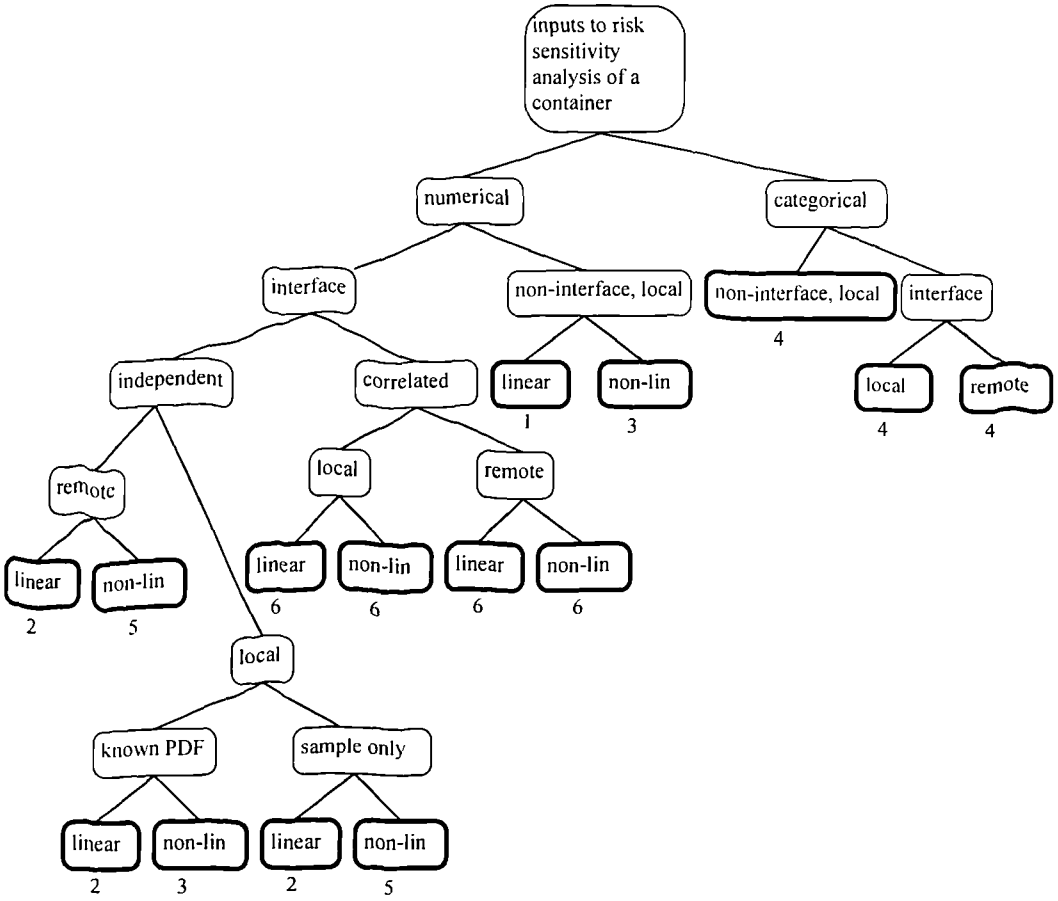


Figure 9-7: Categories of input attribute for hierarchical RSA

Type of input rv	(case no.) Method
categorical	(4) direct calculation by partitioning pre-computed samples
numerical interface: correlated	(6) partial correlation coefficient
numerical non-interface local : linear	(1) linear correlation coefficient
numerical non-interface local : non-linear numerical interface local : non-linear, independent, known PDF	(3) direct calculation by approximating to discrete according to PDF
numerical interface: linear, independent	(2) linear correlation coefficient
numerical interface remote : non-linear, independent numerical interface local : non-linear, independent, sample-only	(5) direct calculation by approximating to discrete using fixed percentiles

Table 9-2: Summary of proposed direct calculation method

In Section 11.2 of Chapter 11 worked examples are presented which illustrate the direct calculation method. The major source of inaccuracy in the results obtained using the proposed method is that the partial

correlation coefficient is used wherever there are correlated numerical interface attributes, even when they are not linearly related to the output attribute. Yet the partial correlation coefficient only indicates the variance reduction which can be accounted for by linear regression of the sample.

As a result of this shortcoming in particular, there may be a certain amount of Y's variance which remains unaccounted for by the calculated risk sensitivities for the inputs. It is proposed that this "surplus variance" should be calculated and displayed, since it provides a crude measure of the accuracy of the results.

Despite re-use of samples, the proposed method is also very computationally expensive; the dominant relationship between the number of inputs in the given container object (N) and the calculation time (T) is given below for each method. It is assumed that the number of simulation runs (R) will remain constant - this is necessary to facilitate the re-use of pre-computed samples. It is also assumed that two copies of the samples for numeric interface attributes are stored to disk during the initial simulation - the randomly ordered one and another which is sorted in order of size. The sorted sample is used in calculating percentiles and rank correlations (see Section 9.2.4.2 below), as well as the probabilities for goal values, and the time taken to sort the pre-computed samples thus does not need to be included when considering the estimated calculation time for performing the risk sensitivity analysis.

1,2) Linear correlation coefficient:

$$T \propto N \times R$$

3) Direct calculation by approximating to discrete with K values:

$$T \propto K \times N \times R$$

4) Direct calculation by partitioning pre-computed samples:

T is small compared to other cases

5) Direct calculation by approximating to discrete using K fixed percentiles:

$$T \propto K \times N \times R$$

6) Partial correlation coefficient, where M of the inputs are correlated with each other:

For large M, this case will be dominated by the matrix inversions required for the multi-linear regressions (see Section 9.2.3.2 and Equation 9-2). Matrix inversion by LU decomposition (see for example pp. 31-38 [Press 1986]) is an N^3 process (although slight improvements can be gained with more complex algorithms, for example the Strassen algorithm has order $N^{\log_2 7}$)

$$T \propto R \times M \times (M+1)^3$$

If the alternative algorithm involving the linear correlation coefficient between each pair of correlated variables were to be used instead, then :

$$T \propto {}^{M}C_2 = R \times M \times (M - 1) / 2$$

Thus, for large M, the major contributor to the computational load is calculating the partial correlation coefficients. An important aim in choosing the container objects is therefore to minimise the number of correlated interface attributes.

9.2.4.2 Rank Solution

It is proposed that a second, alternative method for calculating the risk sensitivities should also be implemented. The rank correlation coefficient should be calculated for each $<Y, xi>$ pair, where the xi may be either a numerical leaf attribute or a numerical interface attribute. The necessary samples will already have been stored during the simulation of the whole model and during the first container sub-simulation used in the Direct Calculation Solution, above. Only the numerical variables (not the IAO variables) will appear in the rank risk sensitivity analysis. Unfortunately, there is no coefficient comparable to the partial linear correlation coefficient for rank correlation and this means that where there are significant correlations between interface attributes, the results will be misleading.

This concludes the description of the method adopted for performing risk sensitivity analysis in the risk tool.

9.3 Configuration Modelling Under Uncertainty

In Section 7.2 of Chapter 7, the impact of variants on a vehicle program was described, and in Section 8.8 of Chapter 8, the role of `IVO` (is-a-variant-of) relationships in the design modelling and risk assessment methodology was described. It was explained that information in the `FeatureModel` is used to switch in out parts of the `ComponentModel`, according to the `IVO` relationships which have been defined between parts of the component model, so as to display the correct Bill-of-Materials for the particular `Product` (i.e. the configuration of components and assemblies) of interest. Section 9.3.1, below, describes the algorithm used to achieve this. It was also explained in Section 8.8 of Chapter 8, that the production volume for a `Component` or `Assembly` is often a significant cost driver, and is often highly uncertain. Section 9.3.2, below, describes the algorithm used to calculate the (uncertain) production volume for a `PhysicalObject` from the estimated sales volumes of the different `Products` and `CustomerOptions` on which the `PhysicalObject` is used. (The class definitions for base classes `FeatureModel`, `ProductModel`, `Feature`, `Product`, `CustomerOption` and `IVO` are all given in Appendix D).

9.3.1 Switching In/Out Parts of the Component Model

In this section the algorithm used to "prune" the component model to represent the product of interest, including only the customer options of interest, is described. The algorithm will be illustrated using a simple example, shown in Figure 9-8.

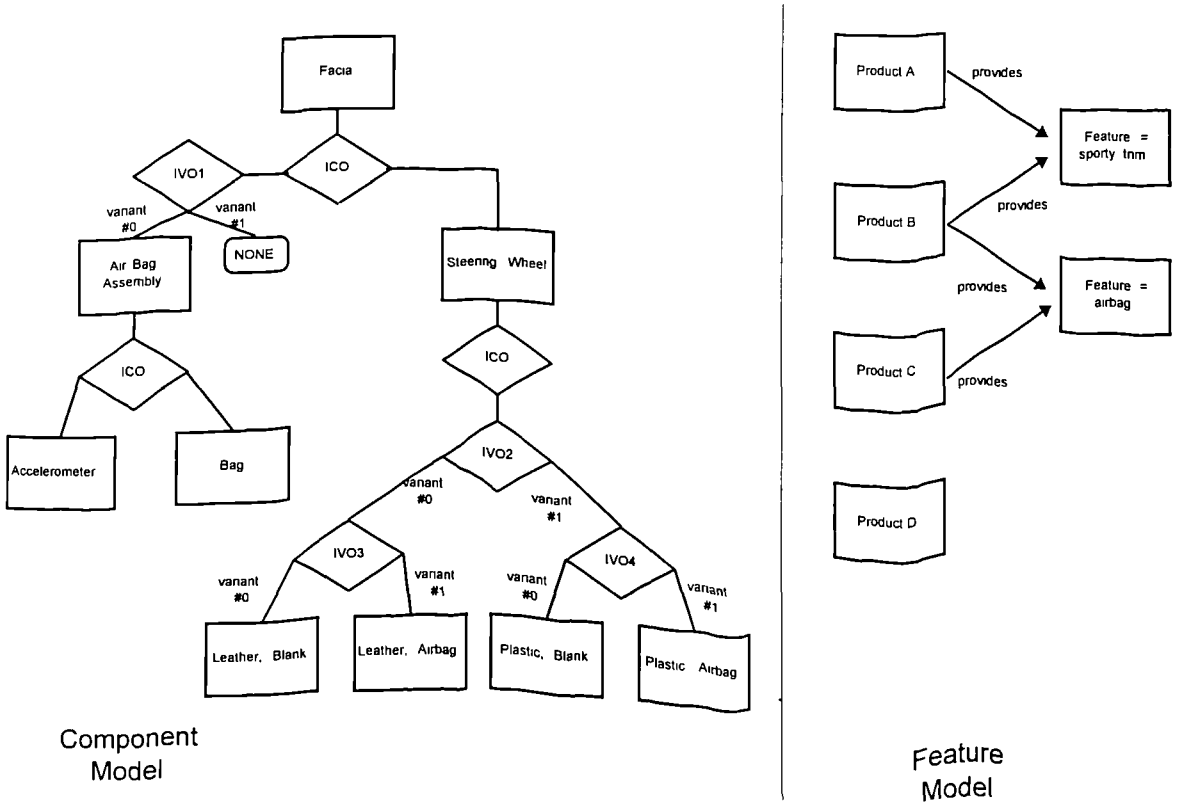


Figure 9-8: A simple example of variants

Consider two features in the Facia area, "sporty trim" and "air-bag", and consider their impact on the steering wheel section of the component model. Suppose that the "sporty trim" feature provides a leather-covered steering wheel as opposed to the plastic-finish one provided usually. And consider that the provision of an air-bag will also affect the construction of the steering wheel centre considerably.

Any object in the component model which has a set of variants (modelled using an `IVO`) is termed a generic object - note that the generic objects have been omitted from Figure 9-8 to simplify the diagram. Product A, which provides the sporty trim (i.e. leather) feature but not an airbag, should make the following selections:

IVO1 = NONE
IVO2 = variant #0
IVO3 = variant #0

Similarly, product B should select:

IVO1 = variant #0
IVO2 = variant #0
IVO3 = variant #1

product C should select:

IVO1 = variant #0
IVO2 = variant #1
IVO4 = variant #1

and product D should select:

IVO1 = NONE
IVO2 = variant #1
IVO4 = variant #0

The mechanism for making these selections is for each feature to store both a list of the IVOs which it affects and also, for each such IVO, the selection which should be made if the feature is provided (as an index into the variant list stored by the IVO). Thus the "sporty trim" feature stores:

IVO2 = variant #0

And the "airbag" feature stores:

IVO1 = variant #0
IVO3 = variant #1
IVO4 = variant #1

The class definition for `Feature` is thus as shown below. The link `implies` is used to allow a hierarchical structuring of the Features in the model - thus if a particular set of features often occur together, a single collective feature can be defined which implies all of them and can then easily be re-used in different parts of the model.

```
class Feature isa EdWithID
 attribute variable affects : unbound IVO[]
 attribute variable selects : INTEGER[]
 attribute variable implies : unbound Feature[]
endclass
```

The example illustrates that each IVO must also have a default selection which is used if none of the features in the product of interest affect the IVO. The mechanism for implementing default selections from IVOs is to add a special feature to a product, and call this special feature the basic model, which selects the defaults. Each product has exactly one basic model - in Figure 9-9 all the products have the same basic model which is called "basic vehicle" - in general, however, there may be more than one basic model available. In the example, the "basic vehicle" feature stores:

IVO1 = NONE
IVO2 = variant #1
IVO3 = variant #0
IVO4 = variant #0

Notice that values are stored for all the IVOs, even though IVO2 = variant#1 excludes IVO3. Thus, if the "sporty trim" feature is added to the "basic vehicle" causing the selection IVO2 = variant #0, the value of IVO3 will be correct.

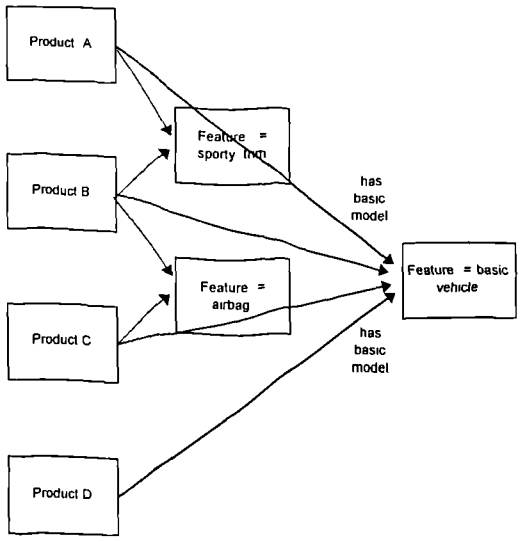


Figure 9-9: Feature model includes a basic model for each product

When the risk modeller chooses a product of interest - for example ProductA - the risk tool must:

1. assign a value of NONE to all selections in all IVOs
2. assign the IVO selections made by the basic model for Product A
3. assign the IVO selections made by each other feature provided by Product A, in turn

If the selection for a particular IVO is assigned two different values during step 3, an error will be flagged because this implies that the product provides inconsistent features. When these three steps have been performed, the component model will have been correctly "pruned" to represent the particular product of interest. Next we consider the customer options. Each product has a list of its valid customer options, and these provide additional features. In the example, the "airbag" feature might be offered as a customer option with product D (see Figure 9-10).

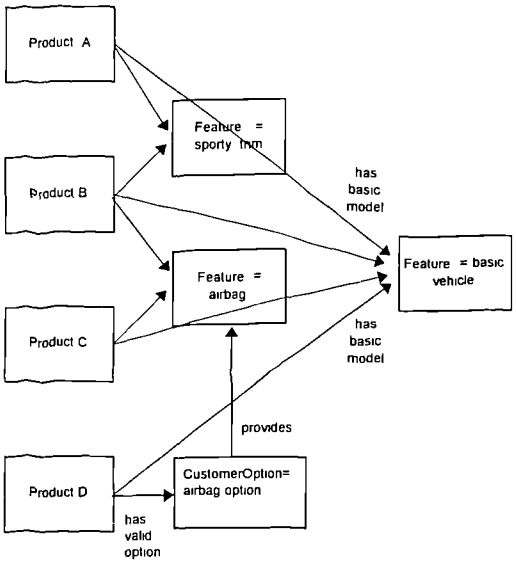


Figure 9-10: Feature model includes "valid options" for each product

When the user specifies a product of interest, prior to evaluating the risk model, they will also choose which of the valid options to include. Thus, when pruning the component model to represent the chosen product, after following steps 1-3 above, the risk tool must follow step 4:

4. Assign the IVO selections made by those valid customer options for the selected product which have been selected (i.e. included) by the user.

If an IVO which had a selection made during step 3. then has another, different, selection made during step 4, then this is permitted even though it may mean that the feature model is inconsistent; i.e. it may not be possible to provide the desired features as well as the selected customer options. On the other hand, an IVO may be assigned a value of NONE by the chosen product and then assigned an actual value by a customer option, and clearly this should be permitted. And conversely, customer options to omit parts of products should also be permitted.

The algorithm, as described so far, shows how the risk tool can assign a value to the `selected_variant` link for each `IVO` in the component model (see class definition below), to reflect the selected product and

customer options. The user interface for the risk tool must then display the `selected_variant` links when the user is browsing the component model.

```
class  IVO isa Relationship
 attribute variable variants : unbound SimWithID[]
 attribute variable selected_variant : unbound SimWithID
endclass
```

It remains to describe how the risk tool can then use the selected variants in an evaluation of the risk model. The most obvious approach is to simply replace every generic object encountered during an evaluation with its selected variant. This would be appropriate when evaluating piece part cost, for example, but when evaluating tooling costs the sum of the tool cost over all variants is required. It is also possible to envisage situations where other rules than selection or aggregation might apply - for example one might wish to use the maximum dimensional value which occurs over all variants when selecting packaging. A flexible approach was therefore adopted, allowing arbitrary rules to be defined for propagating an attribute value over a set of variants. A class is derived from `IVO` which provides specialised methods to calculate the returned attribute value over a set of variants. For example, the class `PhysIVO`, shown below, can be used to relate a generic `PhysicalObject` to its set of variants. The `variants` and `selected_variant` links have been specialised, and methods provided for calculating piece, tool and logistics costs.

```
class  PhysIVO isa IVO
  method piece_cost_fn ( ) : FLOAT
  method tool_cost_fn ( ) : FLOAT
  method logistics_cost_fn ( ) : FLOAT
 attribute variable variants : unbound PhysicalObject[]
 attribute variable selected_variant : unbound PhysicalObject
endclass
```

The generic object will use `PhysIVO::piece_cost_fn()` as its derivation route for `piece_cost`, and similarly for the other attributes. Thus the derivation routes, which are part of the existing underlying methodology for the risk model, allow attributes of the selected variant to be used in an evaluation. A minor addition to the algorithm is required however to allow links of the selected variant to be used. Figure 9-11 shows an example which illustrates the problem.

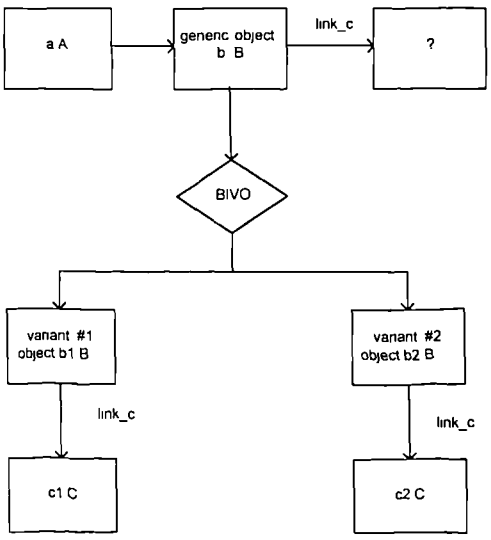


Figure 9-11: Link via an object with variants

Here object *a* follows *link_c* of the generic object *b*, to define the derivation route for one of its own attributes. Clearly what should happen is that *link_c* of the selected variant should be used. Thus *link_c* of object *b* must be set to point to *c1* if variant #1 is selected, or to *c2* if variant #2 is selected. In general, the algorithm when assigning an IVO selection should be as follows:

```
FOR link = each link from selected_variant
        IF link is not of class IVO THEN
                set generic_object.link = selected_variant.link
        ENDIF
NEXT link
```

Where there are nested variants (as in the airbag and steering wheel example described earlier), the assignments should be made from the bottom up. This concludes the description of the algorithm used to "prune" the component model to reflect the selected product and customer options.

9.3.2 Calculating Production Volumes

This section describes the algorithm used to calculate the production volume for each part in the component model. The risk modeller provides a, possibly uncertain, estimate of the sales volume for each product which appears in the feature model, as shown in the class definition below:

```
class  Product isa SimWithID
 attribute variable features : unbound Feature[]
 attribute variable basic_model : unbound Feature
 attribute variable valid_options : bound CustomerOption[]
 attribute variable sales_volume : INTEGER
endclass
```

The risk modeller also provides an estimate (possibly uncertain) of the percentage of products sold which will include a particular option - the `take_up_percentage` in the class definition below:

```
class CustomerOption isa SimWithID
 attribute variable take_up_percentage : FLOAT
 attribute variable include  : BOOL
 attribute variable features : unbound Feature[]
endclass
```

The take-up percentage specifies the sales volume for a customer option. The attribute `include` is used by the risk modeller to signify whether or not this customer option should be included in the actual product instance which is about to be simulated. The component production volumes can be calculated from the sales volumes, provided that a simplifying assumption is made; the assumption that the valid customer options for a particular product are independent. What is meant by independence in this context is that no component provides more than one customer option for a particular product. Consequently, it is adequate to specify the sales volume for each customer option independently in order to calculate the production volumes for the components - it is not necessary to specify the sales volume for each possible combination of customer options. For example, in the case shown in Figure 9-8, it is not permitted to offer both sporty trim and airbag

as customer options on the same product. This would break the independence assumption, since some components (the steering wheel) are affected by both features. Where dependent customer options arise, one of them must be re-defined as a separate product to avoid the problem.

All `PhysicalObjects` have a production volume, `prod_v l`, and may have `variants` defined, as shown in the class definition below:

```
class PhysicalObject isa SimWithID
 attribute variable piece_cost : FLOAT
 attribute variable tool_cost : FLOAT
 attribute variable logistics_cost : FLOAT
 attribute variable prod_vol : INTEGER
 attribute variable vol_coeffs : bound VolCoeffs
 attribute variable variants : bound PhysIVO

 method prod_vol_fn ( ) : INTEGER
 method piece_cost_fn (   : FLOAT
 method tool_cost_fn (   : FLOAT
 method logistics_cost_fn ( ) : FLOAT
endclass
```

`VolCoeffs` (volume coefficients) is used in the calculation of the production volume, as explained below. The algorithm must assign values to the `prod_vol` attributes, using:

- the (possibly uncertain) estimated sales volumes for the products
- the (possibly uncertain) estimated percentage take-up for the customer options
- knowledge of which `PhysicalObjects` are required for each product and customer option
- knowledge of how many of each selected `PhysicalObject` are required (from the cardinalities which appear in the `ICO` relationships)

A simulation of piece cost must be for a particular product and for particular customer option selections within this product. But, in order to simulate one product, one must know what the production volumes will be for all of its components - and since they will be shared with other products, this in turn depends upon the estimated sales volumes of all the other possible products and customer options.

The risk tool must provide a command called "calculate production volumes". When the user executes this command, the tool will use the feature model and the estimated sales volumes for all the products and customer options to calculate production volumes for all `PhysicalObjects`. They are not calculated directly, however, because they may be uncertain values. Instead, each `PhysicalObject` stores a list of coefficients in `VolCoeffs` - one per product in the feature model. The i'th coefficient represents the production volume for the `PhysicalObject` per unit sales volume of the i'th product. Similarly, each `PhysicalObject` will also store one coefficient per product, representing the additional production volume for the `PhysicalObject` per unit sales volume of the product with a customer option selected. The important point here is that, since the customer options must be independent, a `PhysicalObject` can only have its production volume affected by a maximum of one customer option per product. The

`PhysicalObject` will also need to store the identity of the customer option (if any) which influences its production volume. All this information is stored in `VolCoeffs`:

```
class VolCoeffs isa Editable
 attribute variable per_product : INTEGER[]
 attribute variable affected_by_option : unbound CustomerOption[]
 attribute variable extra_per_option : INTEGEP ]
endclass
```

The cardinality of each attribute of `VolCoeffs` is equal to the number of products in the feature model. The independence assumption means that each physical object can only be affected by a maximum of one customer feature per product. Thus, for the i'th product we can store the identity of this customer option (as the pointer `affected_by_option[i]`) and the extra number of components required if the option is included (as `extra_per_option[i]`). The value `per_product[i]` represents the number of components required per unit sale of the i'th product.

During a simulation, the method `prod_vol_fn()` returns the production volume for a particular physical object by evaluating:

$$\sum_{\text{all products}} \text{product.sales\_volume} * (\text{per\_product} + \text{extra\_per\_option} * \text{affected\_by\_option.take\_up\_percentage} * .01)$$

When the user invokes the "calculate production volumes" command, the risk tool must calculate, and store, the values of the `VolCoeffs` lists for all `PhysicalObjects` in the component model. Next, the algorithm for calculating these coefficients is described. Also, since it will be done at the same time, the mechanism for checking that the customer options are independent is described. There are three tasks which must be performed. Firstly, the old instances of `VolCoeffs` must be deleted and new ones instantiated. Secondly, the `affected_by_option` links must be assigned values. This is best combined with the task of checking the independence of the customer options. Thirdly, the values of `per_product` and `extra_per_option` must be assigned. Each of these three tasks is shown in pseudo-code below:

```
FOR phys = first PhysicalObject in component model to last
    IF phys is a generic object THEN
        phys.vol_coeffs = NONE
    ELSE
        delete phys.vol_coeffs
        phys.vol_coeffs = new VolCoeffs(number_of_products)
    END IF
NEXT phys
```

```
FOR product = each product
    FOR customer_option = each customer option on product
        FOR each IVO affected by customer_option
            Proceed down the BOM hierarchy, following ICO, IAO and IVO relationships.
            At every PhysicalObject encountered, assign
             vol_coeffs.affected_by_option = customer_option.
            IF it has already been assigned a different value THEN
                the independence assumption is invalid so flag error and HALT
            END IF
        NEXT IVO
    NEXT customer_option
NEXT product
```

```
FOR i = 0 to (number of products) - 1
    product = i'th product from feature model
    Prune the component model to represent product with no customer options selected.
    CALL top_object.AssignVolCoeffs(i, 1, TRUE)
    Include all valid customer options for product.
    CALL top_object.AssignVolCoeffs(i, 1, FALSE)
NEXT i
```

The algorithm for pruning the component model to represent a particular product is given in Section 9.3.1 "Switching In/Out Parts of the Component Model". `Top_object` is the top object in the component model and `AssignVolCoeffs()` is a recursive C++ method (not a Sim method) defined on classes `PhysicalObject` and `Assembly` - it is a virtual method. The first argument is unchanged as the method recursively calls itself, and represents which coefficient is being assigned a value. The second argument is multiplied by cardinalities (taken from `ICOs`) as the BOM tree (the component model hierarchy) is descended by recursive calls. The second argument represents the number of `PhysicalObjects/Assemblies` required. The third argument indicates whether the method is assigning values to `VolCoeffs::per_product` (if TRUE) or to `VolCoeffs::extra_per_option` (if FALSE). For a `PhysicalObject` with no variants, `AssignVolCoeffs()` can be represented in pseudo-code thus:

```
AssignVolCoeffs(INTEGER i, INTEGER j, BOOL b)
{
    IF b == TRUE THEN
        vol_coeffs.per_product[i] = j
    ELSE
        vol_coeffs.extra_per_option[i] = j - vol_coeffs.per_product[i]
    ENDIF
}
```

Notice that negative values could be assigned to `extra_per_option`, if the customer options switch out parts of the component model. For a `PhysicalObject` or an `Assembly` with variants, the pseudo-code is simply:

```
AssignVolCoeffs(INTEGER i, INTEGER j, BOOL b)
{
    variants.selected_variant.AssignVolCoeffs(i, j, b)
}
```

Thus, the `vol_coeffs` object isn't populated for generic `PhysicalObjects` which have variants: only for those which have no variants. The generic `PhysicalObjects` don't need to know their production volumes - they just obtain their costs from their variants. For an `Assembly` with no variants, `AssignProdValCoeffs` assigns values to the `ICO` children of the object it is invoked upon thus:

```
AssignProdValCoeffs(INTEGER i, INTEGER j, BOOL b)
{
    IF b == TRUE
        vol_coeffs.per_product[i] = j
    ELSE
        vol_coeffs.extra_per_option[i] = j - vol_coeffs.per_product[i]
    ENDIF
    FOR k = 0 TO 1 - number of components in this assembly
        c = cardinality of component[k]
        CALL component[k].AssignVolCoeffs(i, j * c, b)
    NEXT k
}
```

All the algorithms shown above require some adaptation (not shown here in the interest of simplicity) to cope with the possibility of alternatives existing - all alternatives should be assigned the same production value coefficients.

9.4 Summary

In this chapter the computational algorithms used in the risk tool were presented. The choice of Monte Carlo simulation as a means of propagating uncertainty through the risk model was justified, as this is the only technique available which may be used to repeatedly combine arbitrary probability distributions through arbitrary (possibly non-linear, non-monotonic, or discontinuous) functions without an accumulation of error. A linear congruential method augmented with a shuffling routine to break up sequential correlations was selected for generating uniformly distributed pseudo-random numbers. The Latin hypercube sampling method was chosen for generating pseudo-random samples from continuous random variables, as it is more efficient than the rejection method even when the rejection method is used with an aliasing function and is

simpler to implement than proportional allocation. For generating samples from discrete random variables, sampling without replacement provides similar advantages, and was therefore adopted.

The risk sensitivity (RS) of Y to x was defined as the expected value of the reduction in variance of Y when x is replaced by a point value. This definition has the benefit that is applicable to both numerical random variables and also categorical random variables - for example those which occur in the is-an-alternative-of (IAO) relationships in the risk model. A "hierarchical risk sensitivity analysis" was proposed whereby the RS of the output of interest to each input attribute is evaluated at a given level of a "containment" hierarchy in the risk model (based on the Bill-of-Materials hierarchy). The input attribute which is found to make the major contribution to overall uncertainty can then have its own risk sensitivity to each of its inputs explored at a lower level of the containment hierarchy, and so on. The algorithm proposed for evaluating risk sensitivity uses a variety of techniques, depending upon the nature of the input attribute and its relationship to the output and to other inputs. Samples stored during a Monte Carlo simulation of the risk model are re-used wherever possible. The linear correlation coefficient is utilised for numerical random variables (rvs) which are independent and linearly related to the output. Independent numerical rvs which are non-linearly related to the output are approximated to discrete rvs and the expected value of the variance reduction is calculated explicitly by performing a sub-simulation with each discrete value - stored samples for other attributes are re-used during the sub-simulation wherever possible. A method using the partial correlation coefficient is applied when a numerical rv is dependent on other inputs - the effect of the other inputs is effectively removed using the partial correlation coefficient. For categorical variables, the stored sample for Y is partitioned according to the category and the variance of each sub-set is calculated.

The final pair of computational algorithms which were presented in this chapter concern configuration modelling and the representation of variant parts in the risk model. The algorithm used to "prune" the component model to represent the product and customer options specified in the feature model was described - this algorithm automatically assigns the appropraite value to `IVO::selected_variant` for every is-a-variant-of relationship in the risk model. An algorithm was also presented which is used to evaluate a probability distribution for the production volume of each part in the Bill-of-Materials hierarchy. This algorithm takes as its inputs a probability distribution for the sales volume of each product in the feature model and also a distribution for the percentage of customers who are expected to "take-up" each customer option. These inputs are used in conjunction with information stored in the feature and component models about which parts are used for which products/customer options, to automatically calculate the production volume distribution for each part. Production volume has a significant impact on the choice of manufacturing process and hence on cost.

9.5 References

[Deák 1981] Deák, I. "An economical method for random number generation and a normal generator", *Computing*, vol. 27, pp. 113-121, 1981.

[Hays 1970] Hays, W. L. and Winkler, R. L, "*Statistics: Probability, Inference and Decision, Vol II*", ISBN: 03-084429-0, Holt, Reinhart and Winston Inc., New York, 1970.

[Iman 1980] Iman, R. L., Davenport, J. M. and Zeigler, D. K., "Latin hypercube sampling (program users guide)", prepared by Sandia Laboratories for the US Department of Energy under contract DE-AC04-76DP00789, *SAND79-1473*, 1980.

[Knuth 1981] Knuth, D. E., "*The Art of Computer Programming: Volume 2 Seminumerical Algorithms*", Addison-Wesley: Reading, Massachusetts, 1981.

[Law 1991] Law, A. M. and Kelton, W. D., "*Simulation Modeling and Analysis*", Second Edition, ISBN 0-07-036698-5, McGraw-Hill Inc., New York, 1991.

[Mauro 1986] Mauro, C. A., "Efficient Identification of Important Factors in Large Scale Simulations", *Proceedings 1986 Winter Simulation Conference*, Washington D. C., pp 296-305, 1986.

[McKay 1979] McKay, M. D. and Beckman, R. J., "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code", *Technometrics*, vol. 21, no. 2, pp. 239-245, May 1979.

[Press 1986] Press, W. H., Flannery, B. P.,Teukolsky S. A. and Vetterling, W. T., "*Numerical Recipes: The Art of Scientific Computing*", Cambridge University Press: Cambridge, 1986.

[Rao 1973] Rao, C. R, "*Linear Statistical Inference and Its Applications: Second Edition*", ISBN 0-471-70823-2, John Wiley and Sons, New York, 1973.

[Schruben 1986] Schruben, L. W., "Simulation Optimization Using Frequency Domain Methods", *Proceedings 1986 Winter Simulation Conference*, Washington D. C., pp 366-369, 1986.

[Walker 1977] Walker, A. J., "An Efficient Method for Generating Discrete Random Variables with General Distributions", *ACM Transactions on Mathematical Software*, vol. 3, no. 3, pp. 253-256, 1977.

CHAPTER 10

The Risk Tool (RiTo)

A software tool has been written to support the methodology, described in Chapter 8, and using the techniques described in Chapter 9 to evaluate the risk model. In this Chapter the tool itself, which is named RiTo, is described. RiTo uses Latin hypercube sampling and integer sampling without replacement when performing a Monte Carlo simulation of the risk model, and uses the algorithm described in Chapter 9 to represent variant designs (i.e. for configuration modelling under uncertainty). The risk sensitivity analysis algorithms described in Chapter 9 have not been fully implemented and incorporated into RiTo's user interface, however the numerical part of the algorithm has been implemented and tested (and this is described in Chapter 11). The first section provides an overview of how RiTo has been implemented; in particular the structure of the software, choice of platform and programming language, interactions with other software packages and with data files are described. In the second section, an overview of RiTo's capabilities is presented from the perspective of the user. The third section contains some brief comments regarding the design of the software and the use of the Fusion CASE tool during the software design process (Appendix K describes the software design in more detail).

10.1 Overview of RiTo Architecture

RiTo enables a team of designers to build, edit, browse and evaluate a shared OO model of the designed artefact which incorporates uncertainty - a *risk model*. The risk model is stored in an *object repository file* and is built by creating instances of `Editable` (and `Sim`) classes, and editing their link and attribute values. It is assumed that there are two types of user for RiTo; the team of *designers*, who build and edit the risk model, and a single *modeller* who is responsible for writing and maintaining the class definitions. RiTo does not currently provide a graphical user-interface for building and editing the risk model; the designers must use a text editor to modify the object repository file directly. The *modeller* can define new classes of `Sim` object for use in the risk model using the graphical modelling interface provided by the FusionCASE CASE tool (from SoftCASE consulting). Alternatively, the modeller may define the classes directly by editing the *schema file* which stores the class definitions. The file formats for the object repository file and the schema file are given in Appendix E.

Once the risk model has been built, designers may invoke the RiTo user interface executable, load the risk model from the object repository file, browse the information in the risk model, set goal values (for example budgets) for attributes in the model and evaluate the model by Monte Carlo simulation. They can obtain graphs and statistics describing the results of the simulation, and generate reports describing the state of the risk model. Some limited modifications to the risk model can be made from the user interface - for example,

re-numbering all objects and automatic calculation of component manufacturing volumes. The facilities provided from the user interface executable are described in Section 10.2.

There is a 1:1 relationship between the `Editable` C++ classes used by RiTo and the classes with which the designers build the risk model - namely, the base classes plus, optionally, any user-defined classes. The base classes are described in Appendix D. The main reason for choosing this architecture, other than code legibility, was that method invocation during Monte Carlo simulation is fast and efficient because the method bodies are directly executable C++. Also, the modeller is free to implement any method which can be built in C++. In principle, for example, methods which invoke other applications could be defined in RiTo. This decision means that some C++ code must be written and compiled when the user-defined classes are modified by the modeller, but a code generator has been written to simplify this task.

RiTo was implemented on a PC platform, running Microsoft Windows 3.1, using Version 1.51 of the Microsoft Visual C++ (VC++) development environment and compiler. The choice of platform and development tools was largely driven by the needs of the collaborating industrial partners. Figure 10-1 provides an overview of the overall architecture in the implementation of RiTo. The process of defining or modifying class definitions may be summarised as follows:

> The modeller defines classes using the Fusion CASE tool, which generates an ASCII text file containing the user-defined class definitions in Fusion syntax. This file is appended to an existing class definition file containing the base classes, to build the new schema file. The code generator then takes as input the schema file and creates or modifies C++ class definitions, where this is necessary, to match the C++ with the schema. The modeller implements the method bodies for the new user-defined C++ classes and the C++ code is then compiled and statically linked to create a dynamically linked library (*DLL*). The designer may then run the user interface executable, which will be *dynamically linked* with the new DLL when it is invoked, and may load, browse and evaluate risk models which include instances of the newly defined classes.

There are three principle software components shown in Figure 10-1 which have been written (all using VC++) as part of this research; RITO.EXE (the user interface executable), SIMDLL.DLL (the DLL) and CODE_GEN.EXE (the code generator). RITO.EXE is a 16-bit Windows 3.1 application, which thus may be used with Windows 3.1, Windows95 or Windows NT. Two commercial C++ class libraries are used by RITO.EXE; the Microsoft Foundation Classes (MFC) class library of graphical user interface components (which is supplied with VC++) and the Graphics Server 4 class library of graph and chart plotting components from Bits Per Second. RITO.EXE provides the user-interface which the designers use to browse and evaluate the risk model. It is designed using the Document-View application architecture prescribed by the MFC and is compliant with the Windows 3.1 user-interface paradigm, including, for example, on-line context-sensitive Help. It provides a "Multiple Document Interface" (MDI) meaning that multiple risk models may be opened simultaneously, and it provides multiple views onto each risk model. This application architecture is well documented and familiar to any VC++ programmer and the internal design of RITO.EXE is therefore not further described in this thesis.

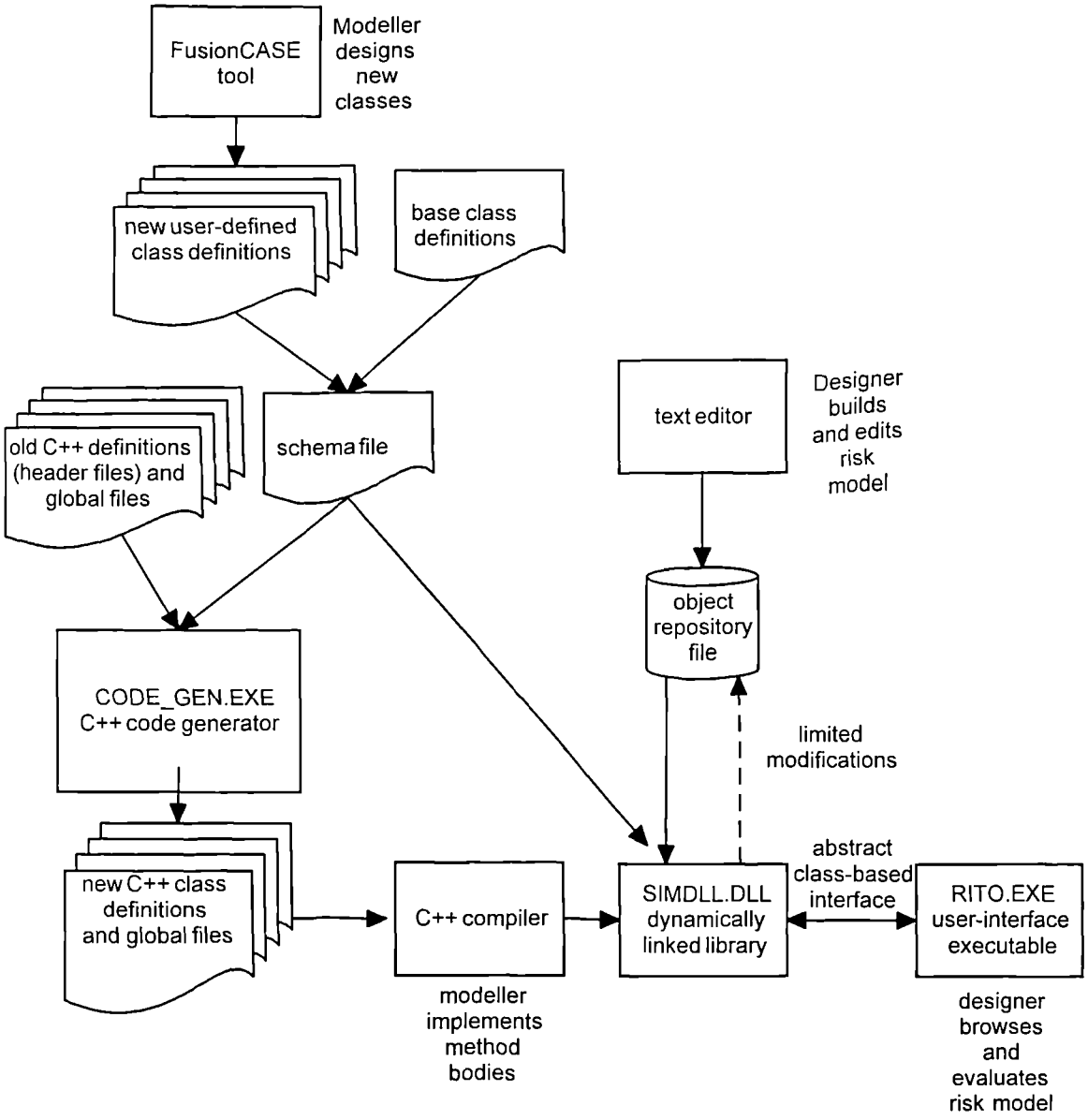


Figure 10-1: Simplified view of data-flow in RiTo implementation

The second software component, SIMDLL.DLL, is a Windows dynamically linked library (*DLL*) which provides the Monte Carlo simulation engine and which also includes the base classes and the user-defined classes. A library of C routines obtained from The University of Texas (named DCDFLIB, see [Brown 1994]) is linked with SIMDLL and this library provides the cumulative distribution functions and their inverses for those continuous PDFs which are supported (other than those which are piece-wise linear). SIMDLL.DLL is linked dynamically (i.e. at run-time) to RITO.EXE. *Dynamic linking* means that the DLL can be modified after RITO.EXE has been compiled and *statically linked*, as shown in Figure 10-1. The third software component, CODE_GEN.EXE, is a C++ code generator which takes the schema file as input and automatically generates the C++ code necessary to implement the classes described in the schema. The code generator creates or modifies C++ headers (class definitions) for each class in the schema file, and also modifies a small number of global C++ files, for example a file named IDS.H which contains a list of unique class identifier constants for classes derived from Sim. The C++ code created by the code generator automatically provides object persistence, run-time schema querying and support for the risk modelling methodology described in Chapter 8 (e.g. multiple derivation routes and uncertain values for each attribute value, alternative link values, etc.). However, the modeller must implement the method bodies by hand.

A class-based interface has been defined between SIMDLL.DLL and RITO.EXE (as opposed to the functional interface which is more usually defined between a DLL and its client), thus supporting a fully object-oriented architecture. A set of abstract C++ classes have been defined which constitute the interface and their class definitions are statically linked with both RITO.EXE and SIMDLL.DLL. The DLL implements exactly one immediate sub-class (i.e. which inherits directly) for each interface class (although there may be many indirect sub-classes). The executable makes calls to the DLL in one of three ways; by creating an instance of the immediate sub-class, by destroying an instance of the immediate sub-class or by invoking a (pure virtual) method defined in the interface class, on an instance of the immediate sub-class. The names of the abstract interface classes all consist of their immediate sub-class name with the post-fix _SD (which is an abbreviation for SimDLL). An interface classes Editable_SD is defined for example, and the DLL implements an immediate sub-class Editable (which inherits directly from Editable_SD). The risk model itself resides in memory which "belongs" to the DLL - the executable provides the GUI-specific implementation, but the DLL provides all the basic functionality for building, editing and evaluating the risk-model and is not Windows-specific. The class-based interface outlined above is described in more detail in Appendix K.

RiTo supports a limited but useful form of schema evolution. If the modeller adds new attributes or links to an existing class definition, and instances of the class were already stored in an existing risk model in the object repository, then the designer may load the risk model into RITO.EXE and re-save it. The new attributes or links will automatically be added and will be assigned a value of UNKNOWN. Similarly, if the modeller deletes existing attributes or links in a class which is already in use then, when the designer loads the risk model into RITO.EXE, the now defunct values will be ignored. Re-naming of attributes and links can generally be achieved simply by using search-and-replace in a text editor.

In the next section, an overview is given of the user-interface presented to the designer by RITO.EXE.

10.2 RiTo's User Interface

10.2.1 Loading a risk model

The designer invokes RiTo by clicking on an icon in the Windows desktop. To load a risk model into memory, the designer selects File from the menu bar, selects Open from the drop down menu which appears and then chooses an object repository file. The view shown in Figure 10-2 is then be displayed - termed the Output view since it includes the Outputs selected for the risk model. The designer may open more than one risk model simultaneously, for example representing different stages of the same design, in order to compare the results obtained from each.

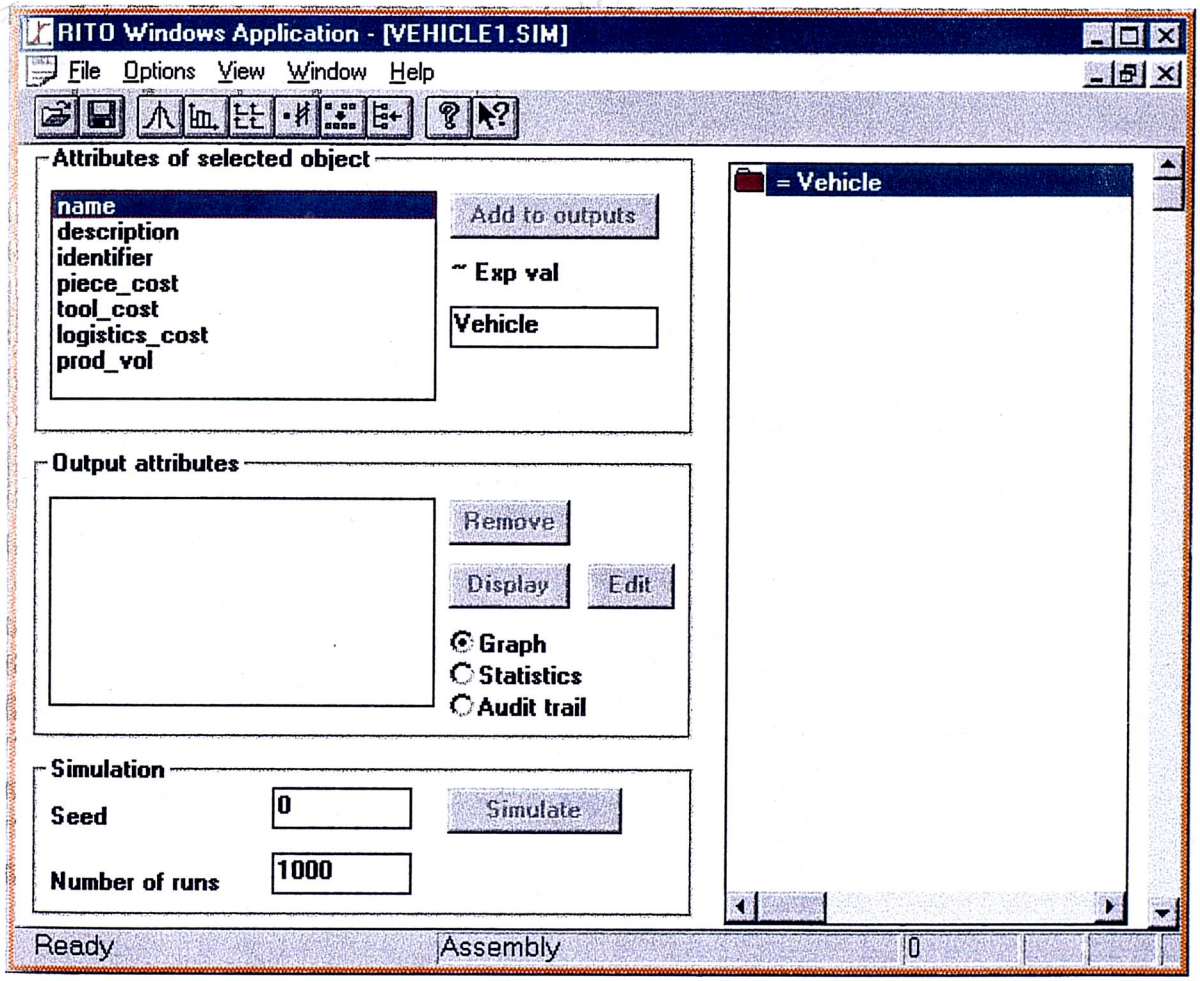


Figure 10-2: Initial Output view displayed by RiTo after loading a risk model

10.2.2 Browsing a risk model

10.2.2.1 The link tree

The large window on the right of the Output view (Figure 10-2) contains an indented list which may be used to navigate the model and is termed the *link tree* in what follows. Each entry in the list represents a link to an Editable object in the risk model (for example a Sim object or an IAO). If the designer double-clicks on the first entry in the list (the top object), it will "open" to show the objects to which the top object has links. Each of these may then be opened and so on.

In a link tree entry, the name of the link is shown, followed by an "=" symbol and then the name of the Editable object. If the object has class derived from SimWithID then the name displayed is simply the value of the attribute "name" and the symbol shown in the indented list is a plain coloured folder. The colour of the folder indicates the class of the object. If the object class is derived from EditableWithID (and not from SimWithID) then the symbol shown is a yellow bag of blocks (indicating that it is a low-level building block object). If the object does not have a "name" attribute, then the name of the class is displayed instead.

If the object represents an ICO (is-a-component-of) relationship then the name displayed is the name of the first component followed by "AND..." and the symbol displayed is a yellow folder with a red ∧ symbol. If the object represents an IAO (is-alternative-of) relationship then the name displayed is the name of the first component followed by "OR..." and the symbol displayed is a yellow folder with a red ∨ symbol. If the object represents an IVO (is-a -variant-of) relationship then the name displayed is the name of the first variant followed by "VAR..." and the symbol displayed is a red folder with a yellow ∨ symbol. Links which are UNKNOWN are shown as a red question mark symbol and links which are NONE are shown as a solid black circle symbol. The status bar at the bottom of the Output view shows the object id of the Editable object currently selected in the link tree, and also the name of the class to which it belongs.

There are two alternative views of the link tree: the view which is initially displayed and is described above as the complete link tree, showing every link of every object. Alternatively a simplified "bill-of-materials" view (BoM view) can be obtained by selecting "View" from the main menu bar and then choosing "Show BOM tree" from the drop-down menu. In the complete link tree, clicking on object A will reveal object B if A has a link to B. In the BoM link tree, clicking on object A will reveal object B if B is a component of A i.e. if there is an ICO object relating A and B. Thus the ICO objects themselves and all objects which do not directly take part in an ICO relationship, are hidden in the BoM view. If the risk model contains variants, there is another important difference between the BoM tree and the complete link tree. In the BoM tree, whenever an IVO object is encountered, only the selected variant is shown, whereas in the complete link tree, all the variants are shown. The designer may change the selected variant by editing the selected product and/or its customer options, using the button on the toolbar (described in 10.2.4 below). The BoM view helps the designer to find a part of the risk model to look at in more detail.

Figure 10-3 shows on the left a complete link tree and on the right, the simplified BoM view of the tree. To switch back to the complete link tree, the designer selects "View" from the main menu bar and then chooses "Show link tree" from the drop-down menu. When the designer toggles between the complete tree and the BoM tree (or vice-versa) , the new view shows the currently selected object at the top of the window. In order to browse above that object, the designer must select "View" from the main menu bar and then choose "Return to Top Object" to re-gain access to the whole risk model. Another way to obtain a new view showing the currently selected object at the top of the link tree is to use the New Window Command on the Window menu; the designer may open as many windows as required on to a single risk

model, each may have a different object at the top, and some may show the complete view of the tree and others the BoM view.

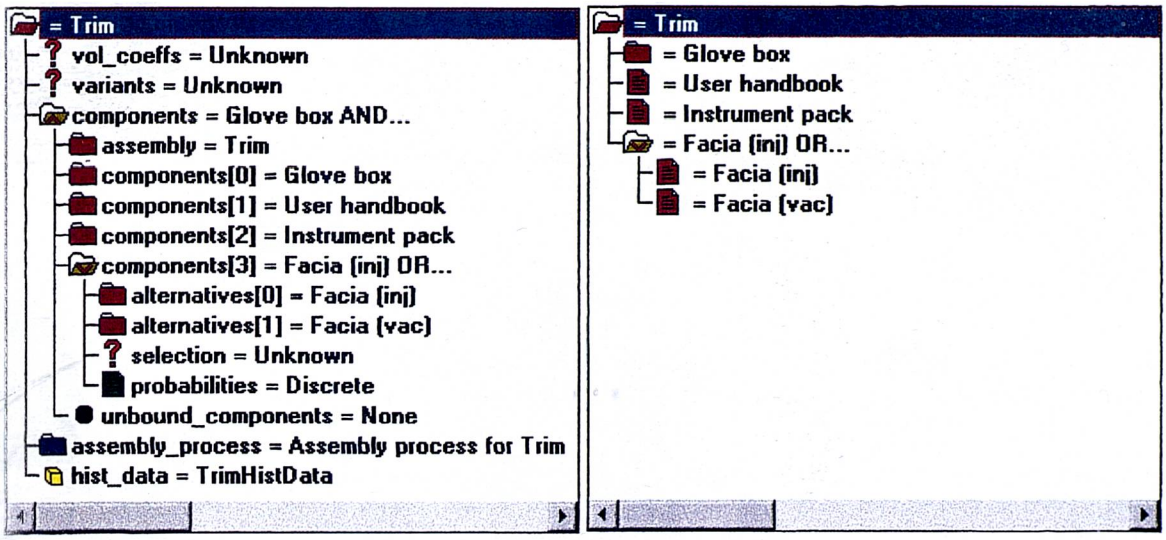


Figure 10-3: Comparing complete link tree with BoM view

The designer may also use the Show None Links and Show Unknown Links commands, on the View menu, to hide/reveal NONE/UNKNOWN links in the link tree - where there are many links which are not used it is sometimes easier to understand the link tree if they are hidden.

10.2.2.2 The "Attributes of selected object" control group

The group of controls labelled "Attributes of selected object" in Figure 10-2 refer to the currently highlighted item in the link tree. For any object in the risk model, its *links* (object-valued) are shown in the link tree and its *attributes* (base-type valued) are shown in the "Attributes of selected object" list box. The edit box labelled "~Exp val" shows an indicative point value for the selected attribute. This can either be an approximation to the expected value (the mean) as shown in Figure 10-2 and indicated by the label "~Exp val" or an approximation to the most likely value (the peak) which is indicated by a label "~Most lik val"- the designer may toggle between the two settings using the first entry in the Options drop-down menu from the main menu bar, or by using the  toolbar button.

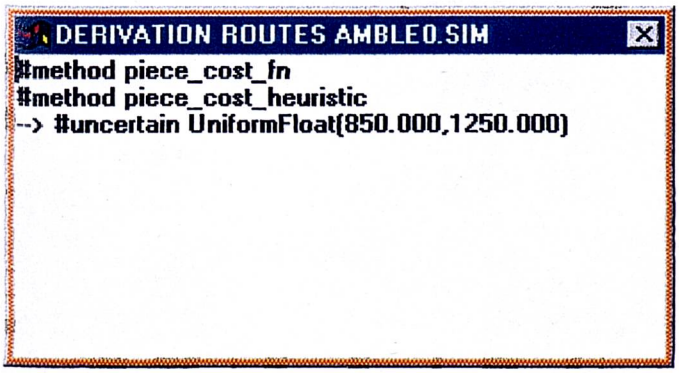


Figure 10-4: The Derivation Routes window

The designer can examine the derivation routes for the selected attribute by selecting the Derivation Routes option from the Window pull-down menu on the main menu bar. The Derivation Routes window will be

displayed, shown in Figure 10-4. As the designer browses the link tree and attribute list, the Derivation Routes window is updated to display the routes for the currently selected attribute. Each derivation route is shown on a separate line, with the most preferred route at the top, and the currently live route (which will be used in an evaluation) is shown with an arrow symbol (-->) to its left. The notation used in the Derivation Routes window is very similar to that used in the object repository file; methods are prefaced with #method, attributes with #att and uncertain values with #uncertain. However, when the derivation route is an UncertainValue object, the displayed information differs from the object repository format because the name of the probability distribution and the values of its parameters are displayed directly. Point values are also displayed directly.

After a simulation has been performed, the probability distributions for the chosen outputs may be examined. To choose an attribute to be an output from a simulation, the designer moves the cursor over the attribute of interest and clicks the "Add to outputs" button.

10.2.2.3 The Audit Trail

When an output has been selected, before a simulation has been performed (or indeed afterwards), the designer may examine its audit trail; this is a textual description of the information in the risk model which will be used when the output is evaluated. More formally, the audit trail is a representation of the attribute derivation network. To view the audit trail the designer selects the output attribute of interest from the list box and then selects the "Audit trail" radio button and clicks on the "Display" button. An indented list is displayed which describes the attribute dependency tree for the selected output attribute. An example is shown in Figure 10-5.

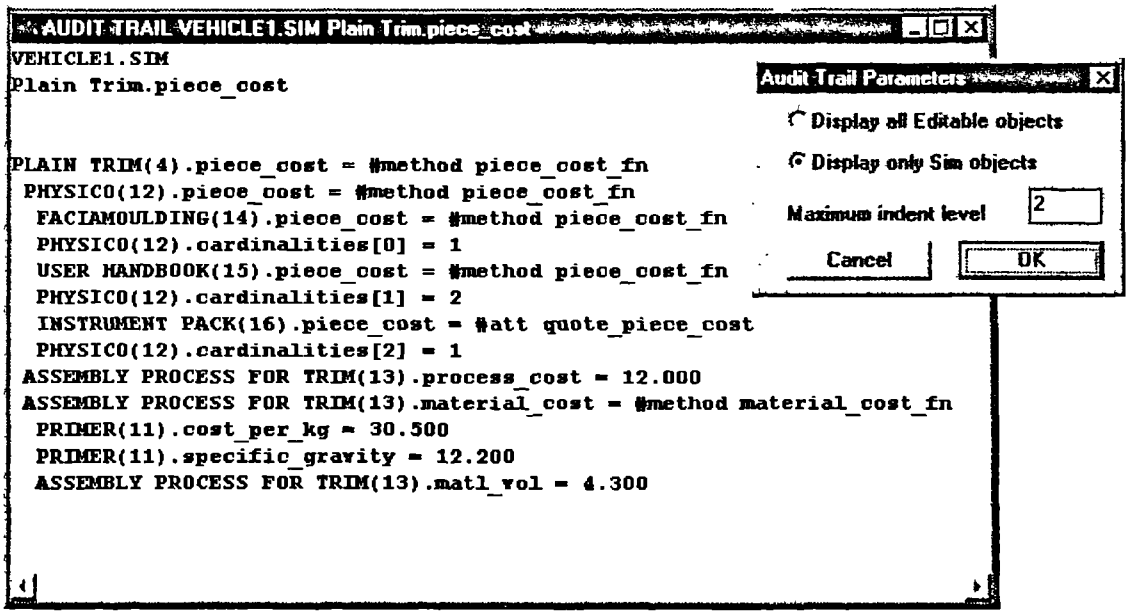


Figure 10-5: An example Audit Trail and the dialog box for editing the Audit Trail parameters

Each line in the indented list represents an attribute value in the risk model. Each attribute has those other attribute values which contribute towards it shown beneath it and indented from it. Only the currently "live routes", which are actually used in the evaluation of the output attribute, are displayed. The format of each line is:

<N spaces> <OBJECT NAME>.<att name>(<object id>)[<att card>] = <derivation route>

The designer may specify the maximum depth of indentation to be displayed. The designer may also choose whether to display attributes of all Editable objects (including, for example, PDF parameters) as part of the audit trail, or to limit the display to attributes of objects which are derived from Sim. Some examples of Audit Trail output are included in Appendix J.

10.2.3 Simulating a risk model and viewing the results

10.2.3.1 The "Simulation" control group

When all the attributes of interest have been added, the designer selects the number of simulation runs to perform and the seed value for the random number generator (or leaves it as its default values) and clicks the "Simulate" button to begin the Monte Carlo simulation.

10.2.3.2 The "Output attributes" control group

After a simulation has been performed, the results may be viewed for each output attribute individually - either as a table of statistics (e.g. mean, standard deviation, etc.) or as a graph (showing a histogram of the generated sample). Results can also be viewed for pairs of attributes - again, either as a table of statistics (correlation coefficients) or as a graph (showing a scatter chart of one attribute plotted against the other). A table of statistics (correlation coefficients) can also be viewed for more than two output attributes, but no graphs are available for three or more output attributes. The list of output attributes is a multiple-selection list box. The designer uses the <Ctrl> or <Shift> keys with the mouse to select more than one output attribute, in the usual way for a Windows multiple selection list box.

10.2.3.3 Viewing results for a single output attribute

To view the statistics for a single output attribute, the designer selects the output attribute of interest from the list box and then selects the "Statistics" radio button and clicks on the "Display" button. A "STATS" dialog box, similar to that shown in Figure 10-6 will appear.

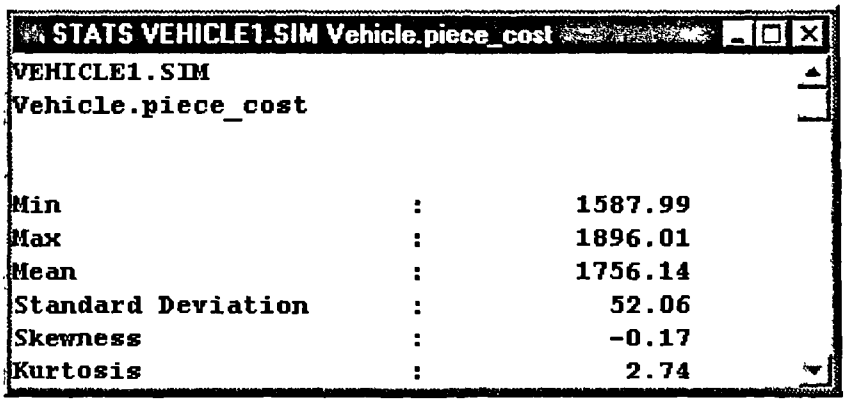


Figure 10-6: Example of an initial STATS dialog box for a single attribute

To copy the statistics to the Windows clipboard, the designer may select the contents of the "STATS" dialog box and press <Ctrl> <C>. If the designer selects the maximise button in the top right hand corner of

the STATS dialog box (or re-sizes the box by dragging the bottom of the frame downwards), the percentile values shown in Figure 10-7 will also become visible. If the designer is interested in percentile values other than 10,..,90 then "goal percentages" can be defined and the next time the statistics are displayed, they will also appear (see 10.2.3.5 below). The designer can also change the number of digits shown after the decimal point using the toolbar button; the setting will affect all floating point values displayed by RiTo.

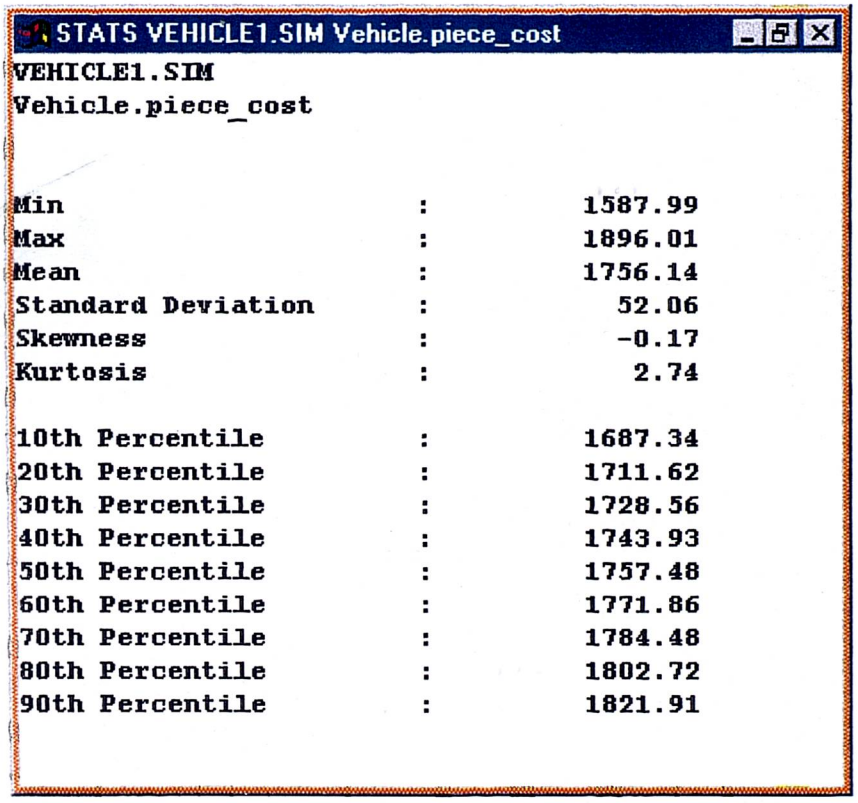


Figure 10-7: Example of an expanded STATS dialog box for a single attribute

To view a histogram of the generated sample for a single output attribute, the designer selects the attribute of interest and then selects the "Graph" radio button and clicks the "Display" button. A histogram similar to that shown in Figure 10-8 should appear.

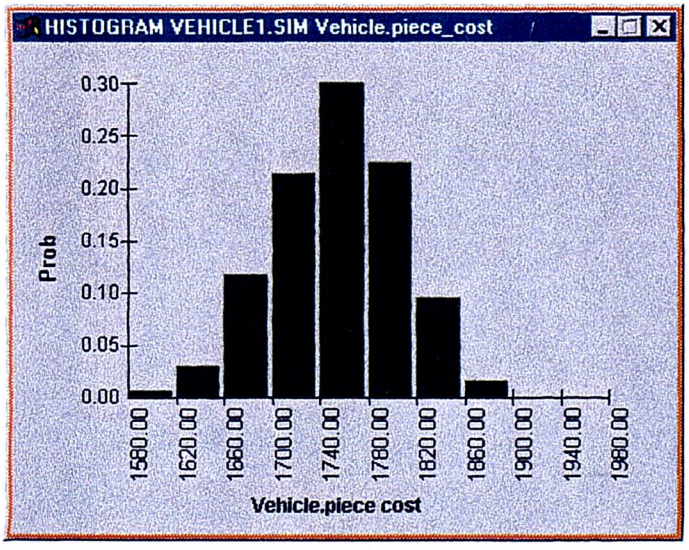


Figure 10-8: Example graph for a single attribute

The height of each bar represents the probability that the random variable will take a value within that bar. Any RiTo graph can be copied onto the Windows clipboard by typing <Ctrl><C>. The graph will be copied in three formats; as an ordinary bitmap, as a device independent bitmap and in Windows Metafile format ("picture").

It is also possible to display the same histogram in Excel; The designer runs Version 5.0 or higher of Excel and loads the file RITODDE.XLS supplied with RiTo. A dialog box should appear in Excel containing the question "This document contains links. Re-establish links?". The designer clicks the "Yes" button, and after a short delay, an Excel chart such as that shown in Figure 10-9 appears, showing a histogram of the currently selected output attribute. If the designer now selects a different output attribute and clicks the "Display" button, the chart shown in Excel will automatically change to show the newly selected output attribute. The data values for the chart are also stored in the Excel spreadsheet and can be cut and pasted from there if required.

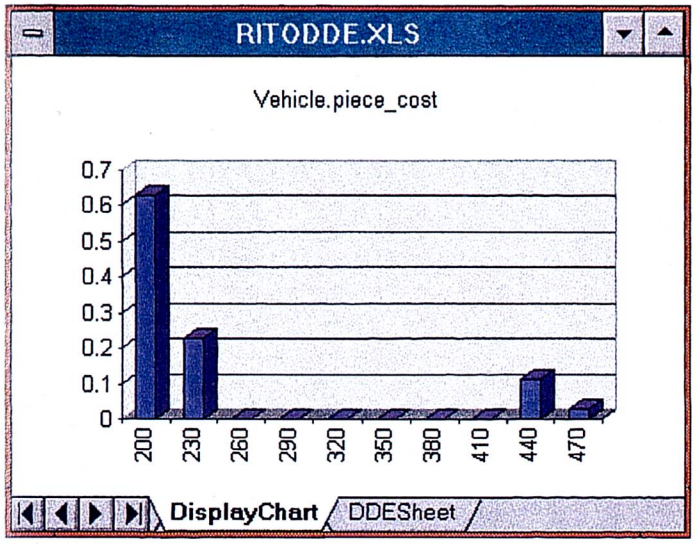


Figure 10-9: An example of an output histogram chart in Excel

The parameters of the histogram can be edited by clicking the "Edit" button when the "Graph" radio button is selected. Clicking this button displays the "Histogram parameters" dialog box shown in Figure 10-10.

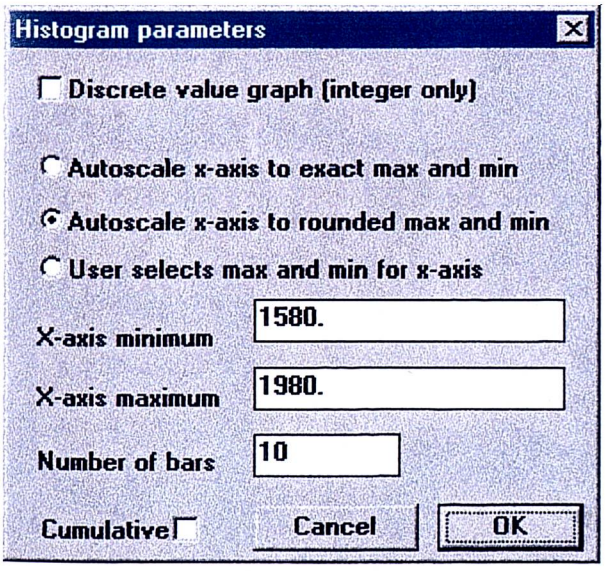


Figure 10-10: Histogram parameters dialog

The number of bars in the histogram may be selected here. If the designer selects the "Cumulative" tick box then a cumulative histogram is displayed. In this case, the height of each bar represents the probability that the random variable takes a value which is less than or equal to the bar's x-axis label. There are three ways in which the maximum and minimum values for the histogram may be chosen. If the "Autoscale x-axis to exact max and min" radio button is selected then the limits of the histogram are the maximum and minimum sample values. If the "Autoscale x-axis to rounded max and min" radio button is selected then the software will choose limits for the histogram so that the maximum and minimum sample values are included but the bar-width is a rounded value - generally such that the x-axis labels only vary by one significant digit. If the "User selects max and min for x-axis" radio button is selected then the user may type whichever values they require into the edit boxes below. This is useful for displaying histograms of one or more attributes on the same scale for comparison. The "Discrete value graph" tick box only applies to attributes of type INTEGER and BOOL; if this is selected, then exactly one bar is displayed for each discrete value which occurs in the sample and the number of bars is automatically set to be the number of discrete values.

There are default histogram parameters which are used whenever a new output sample is generated by simulation. These can be edited by clicking the  button on the toolbar. The "Histogram parameters" dialog is displayed and may be edited as described above. The default histogram parameters apply to all outputs within a particular document (i.e. a particular risk model) - there are separate defaults for each document. The designer may wish to view all the output attributes within a particular risk model on the same scale for comparison. This can be achieved by setting the default histogram parameters to the chosen scale and then resetting all the histograms to the default values using the  button on the toolbar.

10.2.3.4 Viewing results for two or more output attributes

If there are several output attributes of interest, the designer may wish to explore the relationships between them. To view the statistics for two or more output attributes, the designer selects the output attributes of interest from the "Output attributes" list box and then selects the "Statistics" radio button and clicks on the "Display" button. A STATS dialog box, similar to that shown in Figure 10-11 will appear.

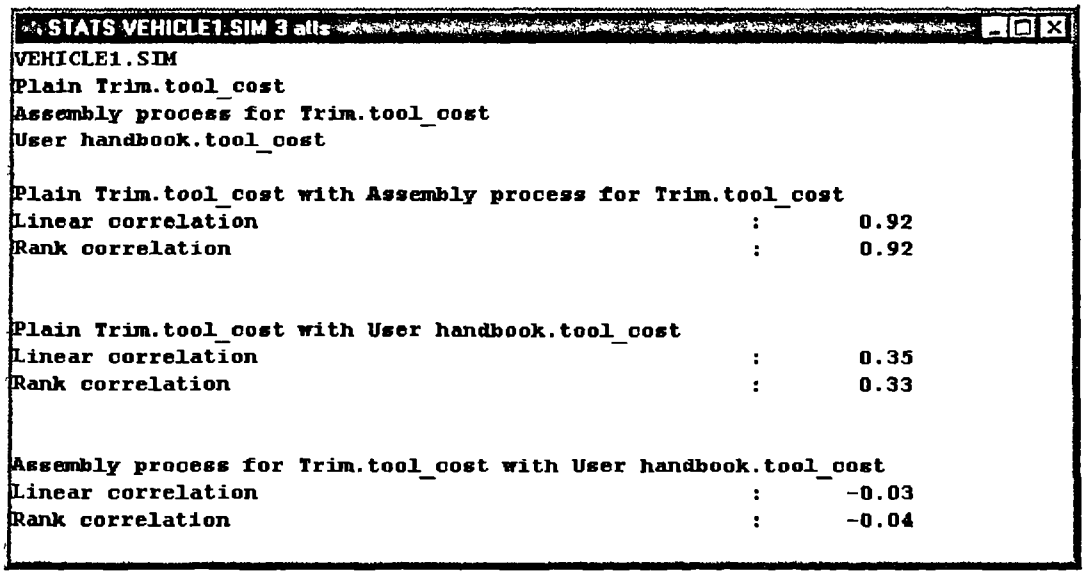


Figure 10-11: Example STATS dialog for more than one output attribute

For each pair of attributes amongst those selected (in this case the three attributes listed at the top of the dialog box) two statistics are given; the *linear correlation coefficient* and the *rank correlation coefficient.*

It is also possible to display a scatter chart showing any each pair of output attributes. To do so, the designer selects the two output attributes of interest, select the "Graph" radio button and clicks on the "Display" button. Figure 10-12 shows the scatter charts for the attributes whose correlation coefficients were shown in Figure 10-11. It can be seen from the correlation coefficients that the user handbook cost is independent of the cost of the assembly process. It can be seen from the scatter charts and the correlation coefficients that the cost of the assembly process is highly linearly correlated with the total cost of the Trim area: if the cost of the assembly process were to be decided (replaced by a point value) then this would lead to a major reduction in uncertainty in the overall trim area costs. This can be quantified using the scatter charts. It is possible to read off from the scatter chart what the minimum and maximum values for the total trim costs would be if the assembly process cost were to be fixed at, say, 842,000 pounds.

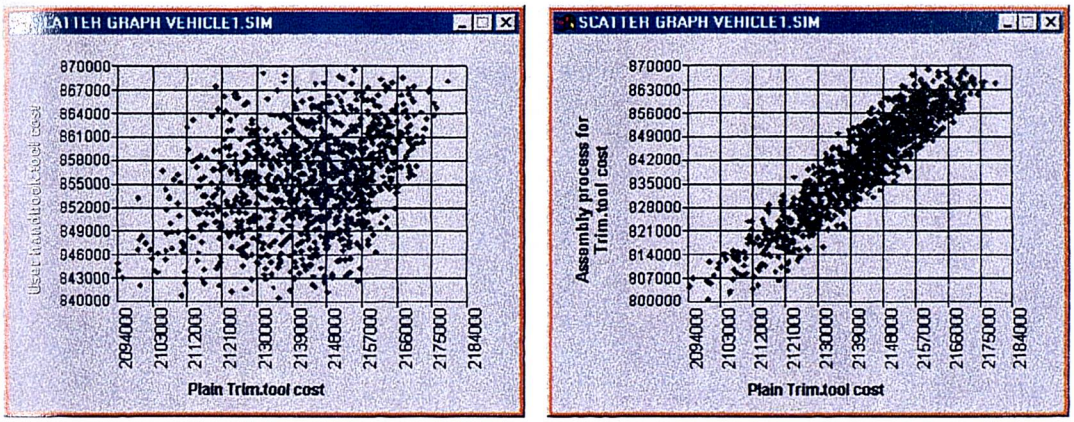


Figure 10-12: Example scatter charts

10.2.3.5 Goal values and percentages

The aim in providing *goal values* and *goal percentages* is to support budget allocation and monitoring. The designer may supply one or more *goal values* for any output attribute. RiTo will calculate the probability (as a percentage value) of failing to achieve these goal values and the information is displayed textually in the "STATS" dialog and also graphically on the histogram or scatter chart. Similarly, the designer may specify one or more *goal probabilities* for any output attribute (as a percentage) and RiTo will calculate the value which has the specified probability of failure to be achieved and display it in the "STATS" dialog and on the graphs.

In order to specify a goal value, a simulation must have already been performed. Goal values can only be defined for attributes of type FLOAT or INTEGER or BOOL. The designer selects the output attribute to which the goal value applies and then selects the "Statistics" radio button and clicks on "Edit". The "Goal Values" dialog shown in Figure 10-13 will then be displayed. The designer types the first goal value into the top edit box and clicks "Add value". This is repeated if there is more than one goal value of interest for this attribute. Conversely, the designer may also specify one or more percentage probabilities (using the second edit box) and RiTo will calculate and display the values which have this probability of being

achieved. The values for percentages 10,20,..,90 (i.e. the 10th to 90th percentiles) are always displayed in the "STATS" dialog for a single attribute.

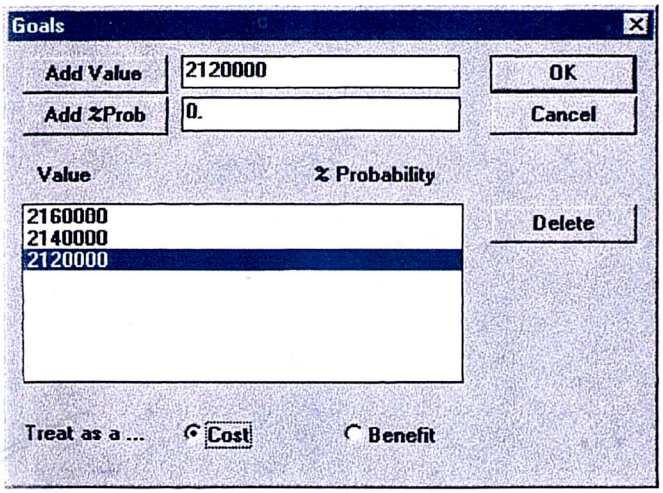


Figure 10-13: Editing goal values and percentages

The "Cost" and "Benefit" radio buttons at the bottom affect the values and probabilities which RiTo calculates and displays in the "STATS" dialog. RiTo always calculates the probability of failure to achieve a goal. Thus, if the designer chooses to treat an attribute as a *cost* then, for each *goal value*, RiTo calculates and displays the probability that the attribute value will be greater than the goal value. Whereas, if the designer chooses to treat an attribute as a *benefit*, then RiTo calculates the probability that the attribute value will be less than or equal to the goal value.

The treatment of goal *probabilities* with respect to cost and benefit is similar; if the designer chooses to treat an attribute as a cost then, for a goal probability P which has been specified, RiTo calculates a value v such that the percentage probability that the attribute is greater than v is P. And, if the designer chooses to treat an attribute as a benefit, then RiTo calculates v where the percentage probability that the attribute is less than or equal to v is P.

Having specified the goal values and percentages, the designer clicks "Display" to see the statistics and then selects the "Graph" radio button and clicks "Display" to see the graph. Example output is shown in Figure 10-15 and Figure 10-15.

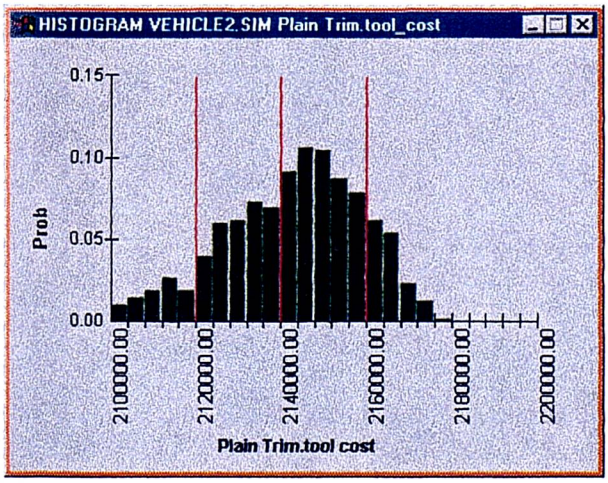


Figure 10-14: Example graphical output with goal values

The goal values are marked in red on the histogram. The last line of the statistics box, for example, indicates that the probability that the tool cost for the trim area will be greater than or equal to £2120K is approximately 91.4 %.

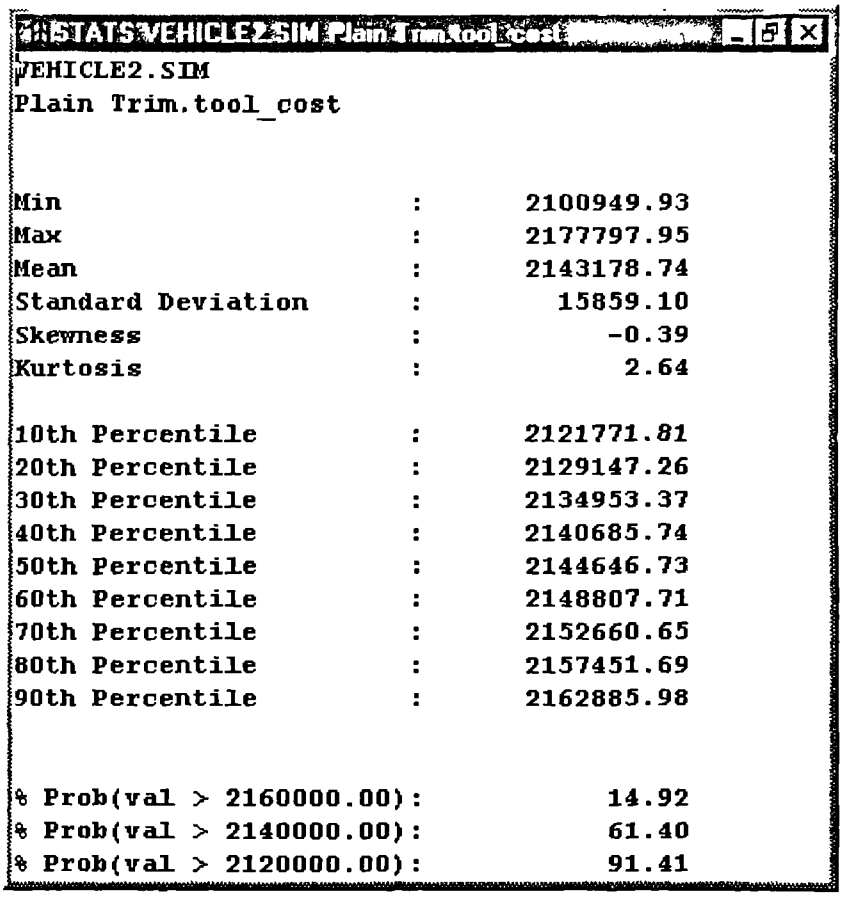


Figure 10-15: Example statistical output with goal values

The designer may also define goal values for two (or more) attributes, and display the probability that the goals will all be achieved simultaneously. Example output is shown in Figure 10-16, where, for example, it can be seen that the probability that the tool cost will be less than or equal to £2120K and the piece cost will be less than £200 is approximately 5.6 %. Notice that the sign of the inequality has been reversed; RiTo always displays the probability of *success* in achieving two or more goals simultaneously.

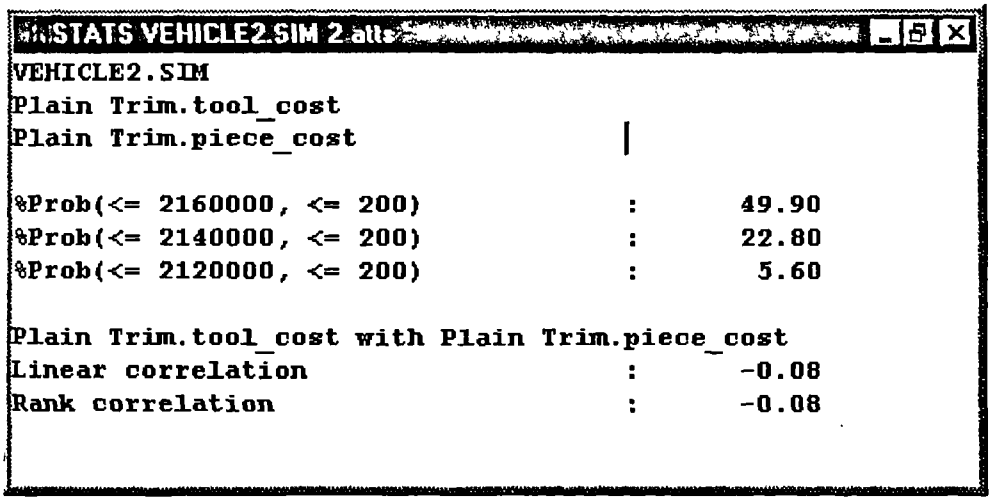


Figure 10-16: Example output with goal values for two attributes

10.2.4 Variant Modelling

If the risk model contains a *feature model* in addition to a component model, then RiTo provides support for selecting which product offering is displayed and evaluated and also for automatic calculation of component production volumes from estimates of sales volumes and customer take-up percentages. The designer may access these features by clicking on the button on the toolbar. The "Edit Feature Model" dialog box, shown in Figure 10-17 ,will be displayed.

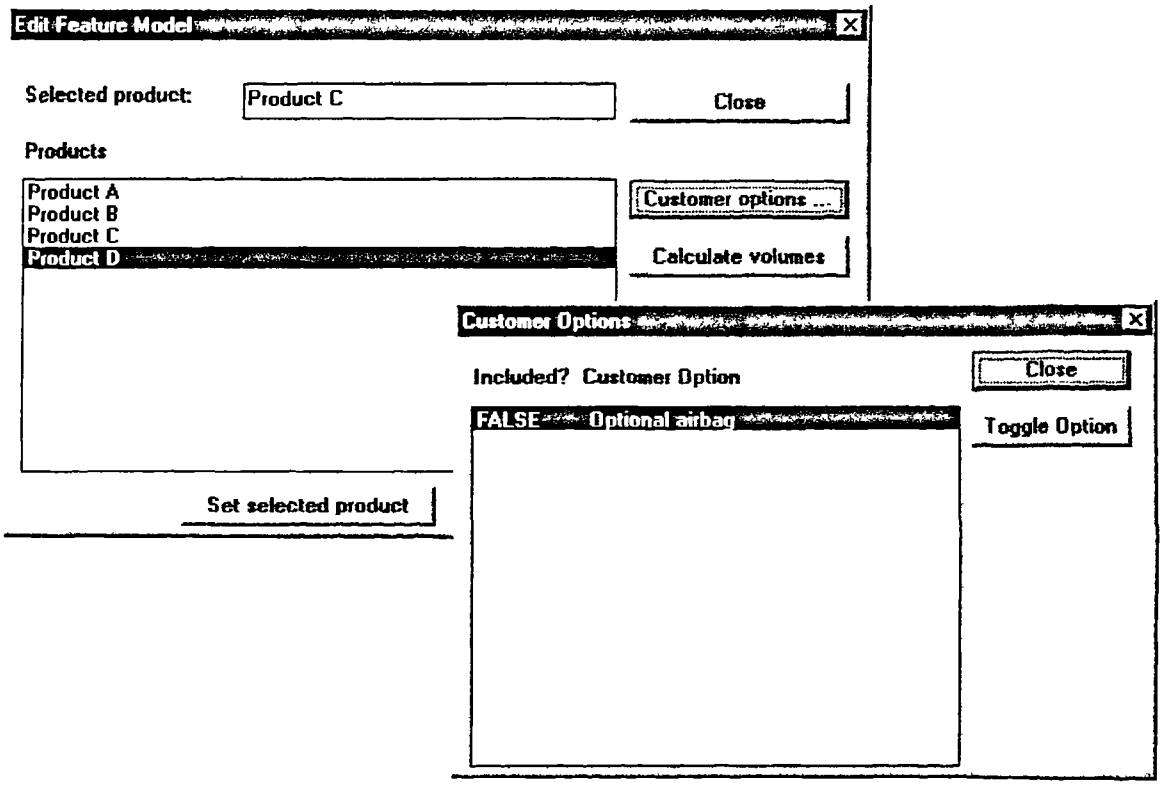


Figure 10-17: The dialog boxes which are used to edit the FeatureModel

This dialog box may be used to:

- Specify which product (from the feature model) the designer wishes to evaluate in the component model.
- Select/deselect customer options for a given product.
- Have RiTo automatically calculate the production volumes for the PhysicalObjects in the component model from the product sales volumes and the customer option take-up percentages stored in the feature model.

The "Selected product" edit box shows the currently selected Product. To change it, the designer chooses the new product required from the "Products" list box and then clicks on the "Set selected product" button. To select/deselect customer options, the designer highlights the product of interest from the "Products" list box and then clicks on the "Customer options ..." button. This will invoke the "Customer options" dialog box, also shown in Figure 10-17. If the button is greyed-out, then the highlighted product has no customer options defined for it. To calculate production volumes, the designer clicks on the "Calculate volumes"

button. All the above new values, including the production volumes, will be written into the risk model and can be saved to disk.

10.2.5 Development of the Risk Model

RiTo allows the designer to load several different risk models simultaneously, and to display graphs and statistics describing them side-by-side. If a "snapshot" of the risk model is stored to disk, with a new filename, at each stage in the design process, it is therefore possible for the designer to gain an overview of the way in which the risk levels are developing over time by displaying a histogram for the attribute of interest from each snapshot.

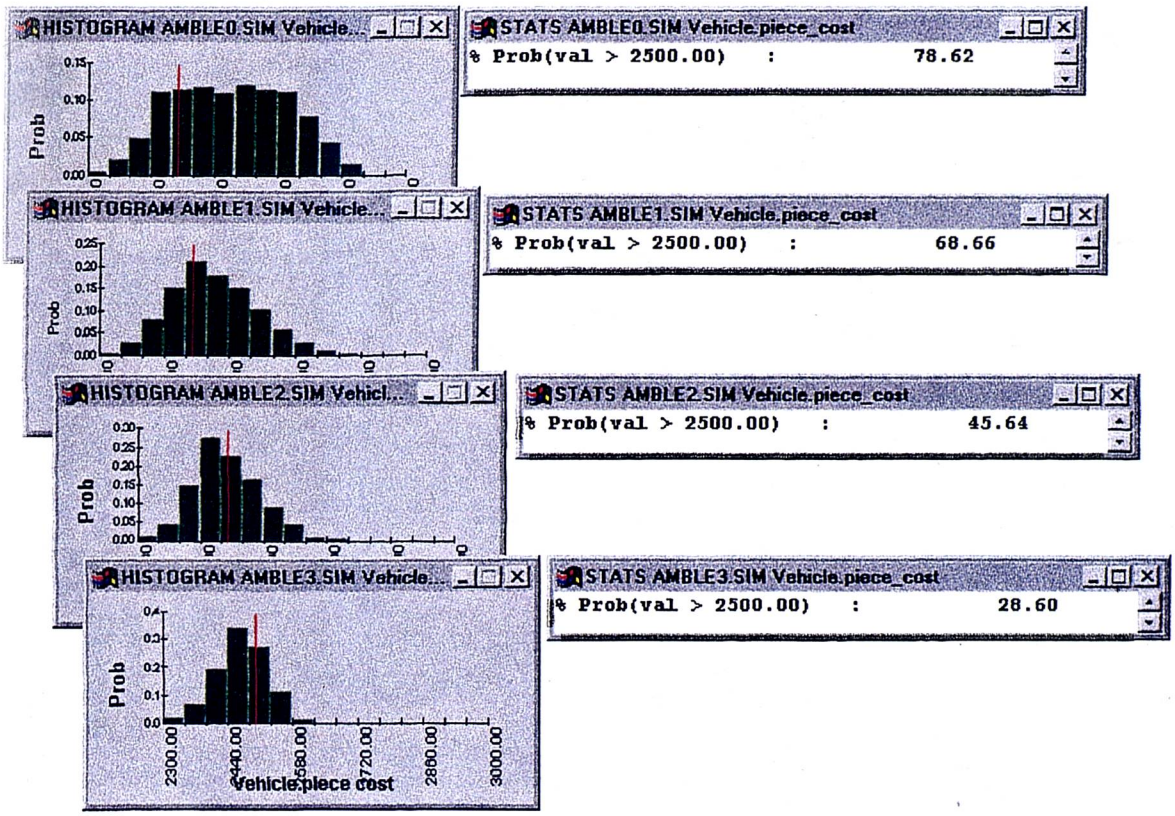


Figure 10-18: Development of the risk model

In Figure 10-18, for example, the histogram for total piece part cost is shown at each of four design stages. The reduction in uncertainty over time can be seen, as the spread of the histogram is gradually reduced at each stage. A budget level of £2,500 is shown marked as a goal value on each histogram, and the statistics boxes next to the histograms show that the probability of exceeding the piece part budget is reducing as the design proceeds.

This concludes an overview of the main features offered to the user from RiTo.

10.3 Some Comments on the Design of RiTo and Use of the CASE Tool

In Section 10.1 the architecture of RiTo was outlined. It was explained that the dynamic linked library SIMDLL.DLL provides the core functionality for the risk toolkit, including the Monte Carlo simulation engine. Appendix K contains a description of the design and implementation issues which arose when building SIMDLL and explains some of the major design decisions which were taken. The Appendix describes, using Fusion graphical views and schemata, some of the C++ classes which were built, and explains in particular how they provide object persistence, run-time schema querying and the ability to evaluate risk models. In this section, some of the key points from the Appendix are summarised, and some comments are also presented regarding the use of the Fusion OO methodology and CASE tool during the software design process.

It is necessary that the `Editable` and `Sim` objects comprising the risk model can be stored in a database and retrieved from it - this is what is meant by *object persistence.* Such an object must also be able to provide information about its own class definition, supplying, for example, the name of its class and the names and types of its attributes, links and methods. This information is given by the class definition stored in the schema and must be determined at run-time (rather than compile-time) - hence a request for such information is termed a *run-time schema query.* Both object persistence and support for run-time queries are provided by the class `Editable`.

As was explained in Section 10.1, a C++ class is defined corresponding to each `Editable` and `Sim` class. There is an instance of a meta-class (`ClassInfo`) held in memory for each `Editable` class, containing the class description. Every instance of class X has a pointer to the `ClassInfo` instance for class X, and this is used to provide the response to run-time schema queries.

New C++ data members, corresponding to the attributes and links given in the Fusion class definition, are not defined directly for each user-defined `Sim` class. Instead, the data members containing the attribute and link values all belong to `Editable` (those containing the attribute derivation routes all belong to `Sim`) and are accessed from derived classes using C++ methods provided by `Editable`. Data members are accessed by index, using integer-valued attribute and link identifiers (IDs) - this being faster than using strings. Static C++ data members (i.e. stored once per class, not once per instance), are defined for each `Sim` class - the static data member has the same name as the attribute and contains the attribute ID. Thus methods defined on user-defined `Sim` classes can utilise these static data members to access attribute values. Object persistence can then be handled entirely by the `Editable` and `Sim` classes, since the data members containing the attribute and link values belong to them.

The design of the evaluation algorithm for the risk model is described in Appendix K. A set of C++ methods termed XFetch methods have been implemented (one for each possible return type) which attempt to "fetch" a given attribute value by recursively following the attribute derivation routes. The same set of methods is used for determining the currently "live" routes in the model, for determining the expected values / most likely values for attributes in the model, for building up lists of random variables prior to a simulation and, when called repeatedly, for fetching sample values during a Monte Carlo simulation. A state flag

representing the current activity in the risk model determines the behaviour of the XFetch methods. The XFetch methods are used to obtain the necessary inputs in user-defined `Sim` method bodies. The random number generation classes are also described in Appendix K - in particular `PDF`, which generates uniformly distributed pseudo-random integers, `ULatin` which provides Latin hypercube sampling to its derived classes, and `UDiscreteWR` which provides integer sampling without replacement to its derived classes.

The graphical views and schemata provided by the Fusion OO methodology have proved useful as a descriptive tool, as shown in Appendix K. However, attempts to use the methodology during the analysis and design phases of the software development met with only limited success. There were several reasons for the difficulties encountered:

Firstly, as currently defined, the methodology seems highly suited to development of applications such as control systems, involving the interaction of many real-world objects with a single "system" - Fusion provides flexible support for the modelling of interactions between multiple agents and the system. However, it is recognised that the current version of Fusion is not well-suited to the development of software systems comprised of multiple re-useable software components combined to form applications - for example, in the Windows environment - where there may often be only one agent (the interface to the user). The creation of such systems is termed domain engineering, and the authors of the Fusion methodology are currently developing the next generation of the Fusion method in which support for domain engineering and modelling of component system architectures is planned - see [Griss 1997] for example.

Secondly, there are useful features of the C++ language which are difficult to represent using Fusion. For example a common mechanism in C++ class libraries is to provide a class (for example `ULatin`, above, for providing Latin Hypercube sampling to a derived class representing a probability distribution) which derived classes then customise by over-riding virtual functions. These virtual functions are invoked by another function of the library class. Thus, for example, a programmer using the `ULatin` class may derive `ULatUniform` from it, over-ride the `InverseCDF()` function and then generate a set of samples from the random variable from elsewhere by creating an instance of `ULatUniform` and then invoking `ULatin::FGetRandomValue`. This is the C++ equivalent of using call-back functions in C. This mechanism is difficult to represent using Fusion, because although the controller class for the `FGetRandomValue` message is `ULatin`, both the object interaction graph and the operation schema will be different for different classes (all derived from `ULatin`) of message recipient. In Fusion, an operation is uniquely defined by the pair <operation name, controller class> but in this case the tuple <operation name, controller class, class of object invoked upon> is required to specify the pre- and post-condition and the pattern of object interaction for the operation. A simple "work-around" for this problem is to define an operation `FGetRandomValue` on `ULatUniform` (although it has no such method) with a suitable operation schema identifying that it does not require implementation. The disadvantage of this work-around is that the functionality of `FGetRandomValue` must be included in the Fusion model once for every class derived from `ULatin`.

The author feels that an understanding of the Fusion methodology and its relationship to a particular implementation language and development environment can only really be gained through using it on a

"real" project. Modelling an existing software development is probably the best way for the analyst/software engineer/programmer to gain this understanding, with "active" use of the methodology reserved until ways of adapting the methodology to suit the particular development environment have been developed. Unfortunately, this was not possible in the research described here, as there was not sufficient time available to gain experience of the methodology prior to commencing development of the risk tool.

Finally, several practical difficulties were encountered in using the CASE tool (due largely to the relative immaturity of the product), some of which were resolved by the end of the project. In the author's experience the use of any formal methodology adds considerably to the time required for software development, and practical difficulties with the CASE tool exacerbated this problem.

10.4 Summary

This chapter has presented a description of RiTo covering its architecture and user interface. Some comments regarding the design of the software and the use of the Fusion case tool were also included.

The architecture chosen for RiTo separates the user interface software module (RITO.EXE) from the software module which is responsible for the creation, editing and evaluation of the risk model (SIMDLL.DLL). The abstract, class-based interface between the two allows them to be modified independently. A graphical Fusion CASE tool may be used to build the user-defined `Sim` class definitions, and a code generator has been developed which translates a Fusion class definition file into C++, which is then compiled into SIMDLL.DLL.

RiTo is a Windows application and provides a user interface typical of Windows 3.1. A "link tree", rather like that provided by the Windows 3.1 File Manager is used to navigate the risk model by following the links between `Sim` objects - multiple views onto the link tree are provided, and there is a choice between the full link tree and a simplified "Bill-of-Materials" view. Attribute values may be displayed, as may indicative point values and derivation routes for attributes, the risk model may be evaluated by Monte Carlo simulation and graphs and statistics describing the results of simulations may be displayed. RiTo also supports the display of an "audit trail", showing all the information in the risk model which was used to obtain a value for an attribute, and the editing and display of goal values and goal probabilities.

10.5 References

[Brown 1994] Brown, B. W., Lovato, J. and Russell, K., "DCDFLIB - Library of of C Routines for Cumulative Distribution Functions, Inverses, and Other Parameters", Department of Biomathematics, The University of Texas, February 1994.

[Griss 1997] Griss, M. L., "Domain Engineering in the Re-use Driven Software Engineering Business: Part II", *Fusion Newsletter*, Hewlett-Packard Laboratories, vol. 5.2, May 1997.

CHAPTER 11

Implementing Models of the Case Studies

In this chapter, risk models based on the case studies described in Chapter 7 are presented - a steel roughing mill and a shipboard dynamic positioning system for Cegelec Projects and the interior trim for a new vehicle development at Rover Group. The way in which the case studies were modelled using the methodology of Chapter 8 is outlined, the classes and object instances in the risk models are described and the simulation results generated by RiTo are presented. In the first section, the simulation results obtained using RiTo for the steel roughing mill case study are compared with those obtained by modelling the case study using a commercially available spreadsheet program with a Monte Carlo simulation engine. This serves as a validation and testing exercise for RiTo. The risk model for the dynamic positioning system is also briefly illustrated in the first section. The second section contains a description of the risk model of interior trim for a new vehicle - this risk model involves both risk sensitivity analysis (identifying the major sources of uncertainty in a risk model) and also configuration modelling under uncertainty (modelling variant parts). The risk sensitivity analysis algorithm (presented in Chapter 9) is explored using a set of examples which span the different circumstances to which the algorithm is applicable and where practical the results are verified using analytical methods or more direct numerical methods.

11.1 Tendering for Large Scale Electrical Installations at Cegelec Projects

11.1.1 Overview of Control Installation Schema

A set of classes have been defined which are specific to control installation design models. The inheritance hierarchy for these classes is shown in Figure 11-1. The classes shown above the dotted line are the RiTo base classes, provided as part of the RiTo tool (see Appendix D for a description of the base classes). Those shown below the dotted line were defined specifically for the control installation design model (see Appendix G for a detailed description of these, Cegelec-specific, classes).

The three most significant classes are `CegSW`, which represents a software component of a control installation design, `CegMan` which represents a management task involved in the design and `CegProduct` which represents a design module. A `CegProduct` may contain hardware (`PhysicalObject`), software (`CegSW`), management (`CegMan`) and other `CegProducts`.

In the base classes, a physical object has attributes which include piece, tooling and logistics cost and all three are aggregated through a hierarchy by the `PhysICO` (physical is-a-component-of) relationship. The additional cost attributes which are introduced in the control installation classes are software cost,

management cost and integration cost, making a total of six types of cost. Hardware cost, introduced in Chapter 4, where this case study was introduced, maps onto piece cost. There is no requirement to model tooling and logistics costs for this particular case study. The meaning of "integration cost" in this context is the cost of integrating hardware and software within a design module.

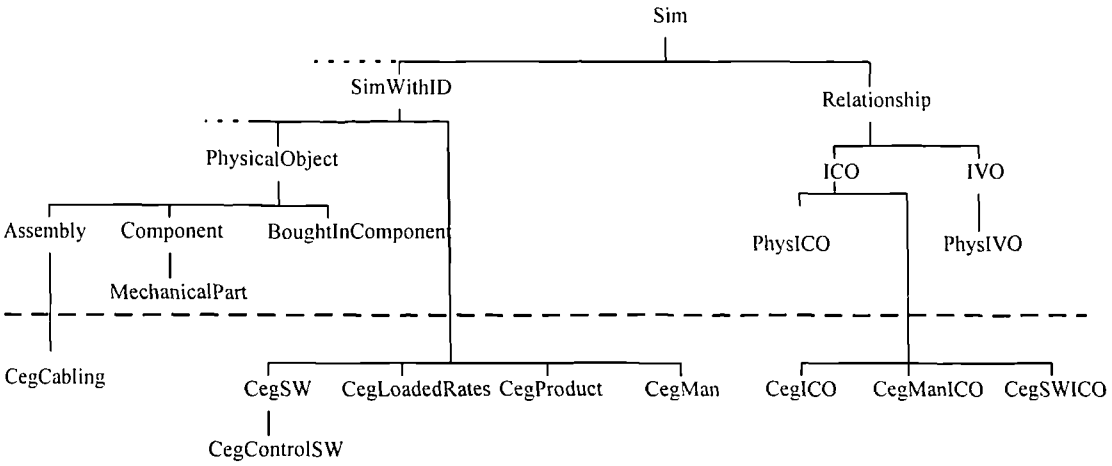


Figure 11-1: Partial inheritance hierarchy for base classes and control installation classes

The `ICO` (is-a-component-of) classes provide the additional methods and data needed to propagate all six types of cost through a heterogeneous hierarchy whose nodes may be hardware, software, management tasks or design modules. Any or all of the six "cost dimensions" can be obtained at any node in the hierarchy, as can the sum of all six cost dimensions. `CegSWICO` relates a single software entity to its component software modules. Similarly `CegManICO` relates a single management entity to its component tasks. `CegICO` relates a single design module (`CegProduct`) to its software (`CegSW`), hardware (`PhysicalObject`), management tasks (`CegMan`) and one or more other design modules (`CegProduct`).

The `CegCabling` and `CegControlSW` classes are provided to support a cross-branch dependency where the piece part cost of the optical cabling depends upon the bandwidth demanded by the control software. The Cegelec-specific classes are described in detail, and their Fusion syntax class definitions are given, in Appendix G.

11.1.2 Spreadsheet Model of the Roughing Mill

To provide a comparison with the RiTo risk model, the roughing mill case study was also modelled using a commercial spreadsheet program with a Monte Carlo simulation engine. The model was implemented as a Microsoft Excel spreadsheet using the "@Risk" extension, from Palisade software. "@Risk" provides cell values which are probability distributions and a choice of a Monte Carlo or a Latin Hypercube simulation engine. The Latin Hypercube option was used in the case study. The spreadsheet model is a hierarchical model and the (uncertain) costs of the nodes in the hierarchy, which are all represented as triangular distributions, are aggregated. Although the containment hierarchy and the numerical values for this particular case study are the same in the spreeadsheet and in the RiTo model, the spreadsheet model is a simplification of the RiTo model in several respects:

1) The RiTo model aggregates different types of cost (piece, tooling, logistics, software, management and integration) independently and obtains values for each at each node as well as providing the local totals. The spreadsheet is a single-dimensional model in the sense that only one attribute (cost) is propagated.

2) The RiTo model would support modelling of manufacturing processes and materials for physical objects using the RiTo base classes. These have been left as NONE to duplicate the information contained in the spreadsheet.

3) The RiTo model would support modelling of alternative designs, but, again, this has not been implemented in order to duplicate the information contained in the spreadsheet.

4) The RiTo model would support modelling of alternative derivation routes for attribute values but this has not been implemented.

A more detailed description of the spreadsheet model may be found in [Crossland 1996].

11.1.3 Object Instances in the Roughing Mill Risk Model

Figure 11-2 shows the simplified, "Bill-of-Materials" view obtained by loading the roughing mill model into RiTo, and Figure 11-3 shows the view obtained by exploring just the first level of the hierarchy. Because Figure 11-3 shows the full link view (as opposed to the simplified view in Figure 11-2), the ICO object named "Control AND ..." is visible which relates the roughing mill to its components. The RoughingMill (a `CegProduct`) contains Management (a instance of `CegMan`), Cabling (an `Assembly`) and Control (a `CegProduct`). The ICO object shown in Figure 11-3 provides a link named `components`, to the components of class `CegProduct` such as Control, a link named `software` to the software components (of which there are none shown here), a link named `hardware` to the components of class `PhysicalObject` such as Cabling and a link named `management` to the components of class `CegMan` such as Management.

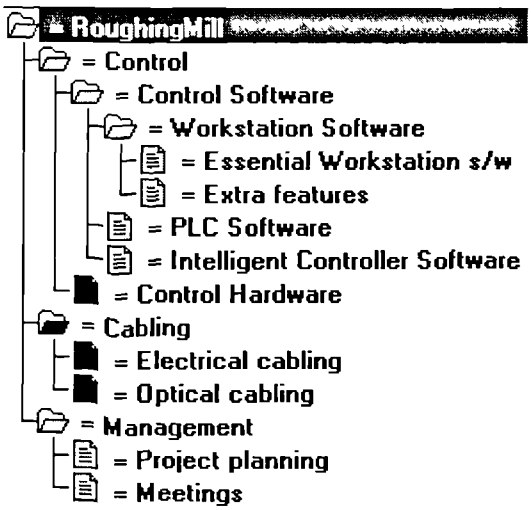


Figure 11-2: "Bill-of-materials" view of RiTo model for roughing mill case study

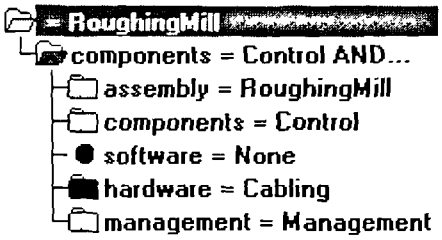


Figure 11-3: Link view of first level of roughing mill case study

Figure 11-4 shows that Cabling consist of OpticalCabling (an instance of `CegCabling` with a link to the control software) and ElectricalCabling (an `Assembly`). Exploring the components of Management (Figure 11-5) it can be seen that it consists of ProjectPlanning and Meetings (both instances of `CegMan`). In order to obtain comparable results to those obtained with the spreadsheet model, it was necessary to use a figure of 1.0 for the weekly costs stored in `LoadedRates`. This is because the spreadsheet model did not include loaded rates - the different types of cost (e.g. management and piece part costs) were not represented or rolled up separately and thus all costs were assumed to have the same units.

The Control branch (Figure 11-6) consist of Control Software (an instance of `CegSW`) and Control Hardware (an `Assembly`). The Control Software is further decomposed into PLC Software, Intelligent Controller Software and WorkstationSoftware (`CegSW` instances). Finally, the Workstation Software is decomposed into Essential Workstation s/w and Extra features (both `CegSW` instances).

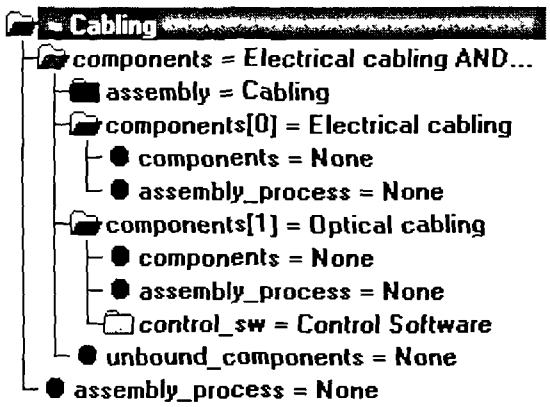


Figure 11-4: Components of Cabling in roughing mill case study

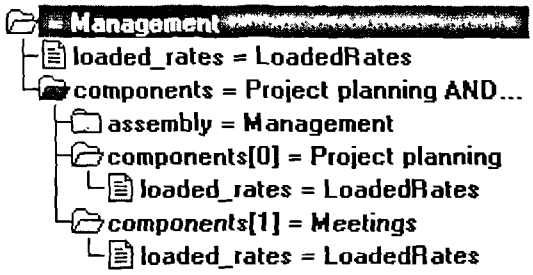


Figure 11-5: Components of Management in roughing mill case study

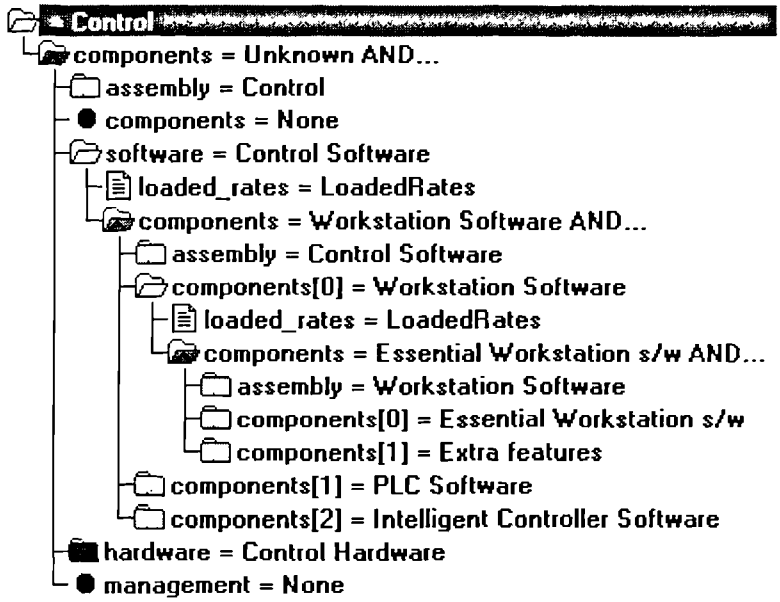


Figure 11-6: The Control branch of the hierarchy in roughing mill case study

11.1.4 Comparison of Results

It was mentioned in Section 10.3.2.3 of Chapter 10 that there are two alternative pseudo-random number generators available in RiTo, either the system-supplied *rand()* function or a method named *UniformRand()* which implements the shuffling algorithm given in Section 9.1.2 of Chapter 9. The programmer chooses between the two options when RiTo is compiled. All the results presented in this chapter were obtained using the system supplied *rand()* function. The *rand()* function was chosen because it was thought that RiTo would execute significantly faster using *rand()* - this was important because the computer used to generate the results presented here had relatively poor performance (it contained a 66MHz 486 processor). It is illustrated in Appendix L, however, that within RiTo the choice of generator has no significant effect on the overall time taken - it is also shown to have no significant effect on the correlations between generated sample values.

11.1.4.1 Indicative Point Values and Distributions for RoughingMill.total_cost

The indicative point value for the total cost of the roughing mill process displayed by @Risk is 407.96. This is the same as that displayed by RiTo when the "Indicate expected values" option has been selected from the main menu. When the "Indicate most likely values" option is selected from the main menu in RiTo, the indicative point value displayed is 370.00 which is close to the distribution maximum, as can be seen from the distributions shown in Figure 11-7 to Figure 11-10 where the histograms generated by RiTo and @Risk for the total cost of the roughing mill are compared. Four simulations were performed with each tool, consisting of 100, 500, 1000 and 5000 iterations. For each simulation both the probability histogram and the cumulative probability histogram are presented. The results obtained using RiTo and @Risk are shown superimposed on the same graph. In each case, a histogram with 20 bars was generated. The limits chosen for the x-axis of the histograms are the minimum sample value obtained using either tool over all four simulations and the maximum sample value obtained using either tool over all four simulations. The value shown beneath each bar of the histograms is the value of the left hand side of the bar.

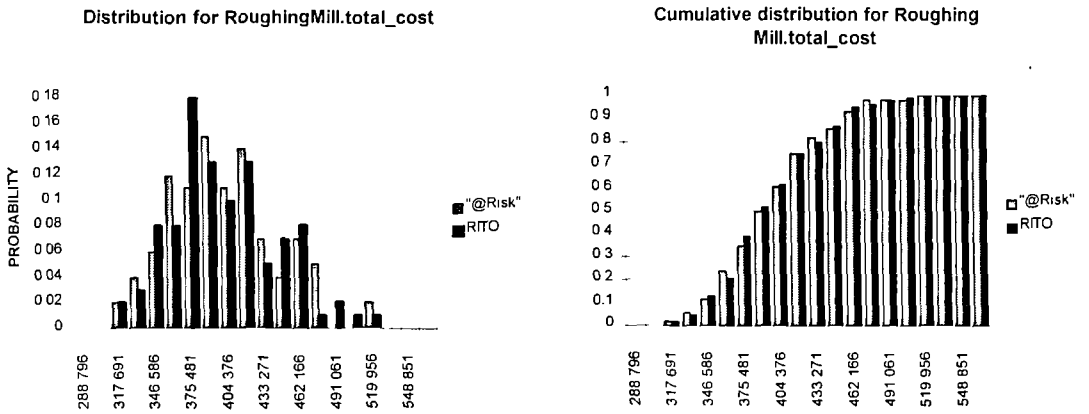


Figure 11-7: 100 iterations

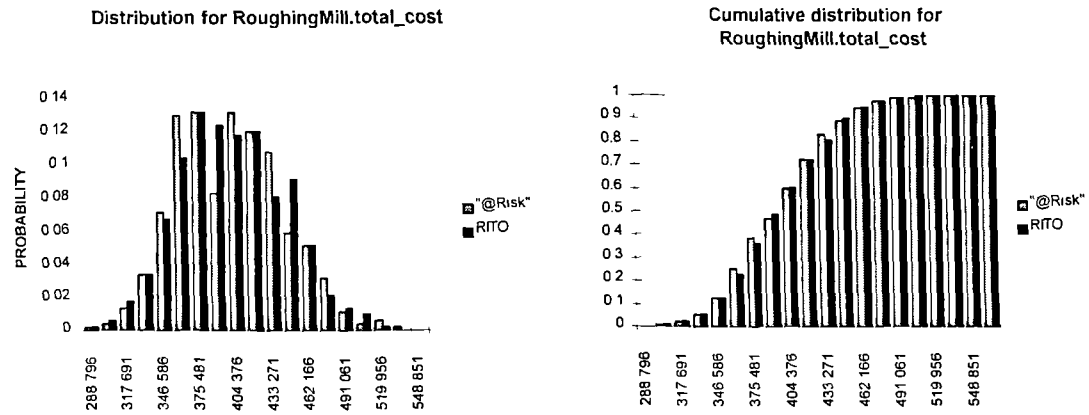


Figure 11-8: 500 iterations

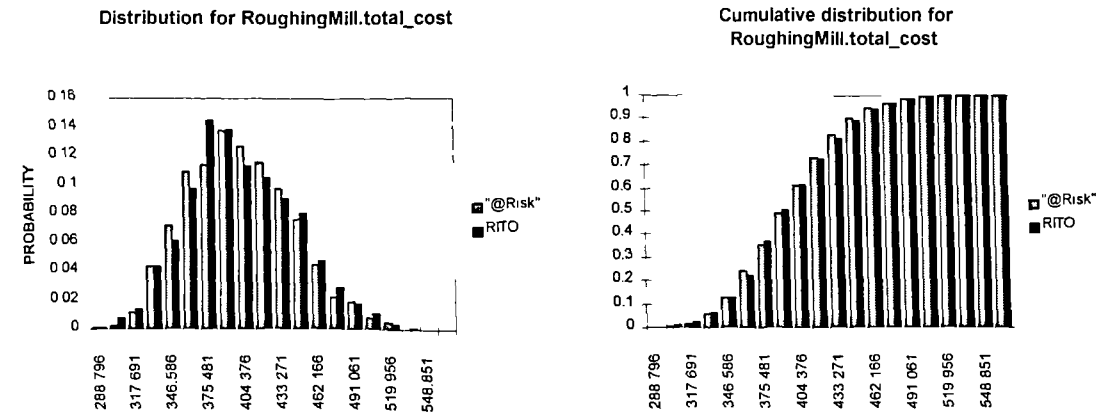


Figure 11-9: 1000 iterations

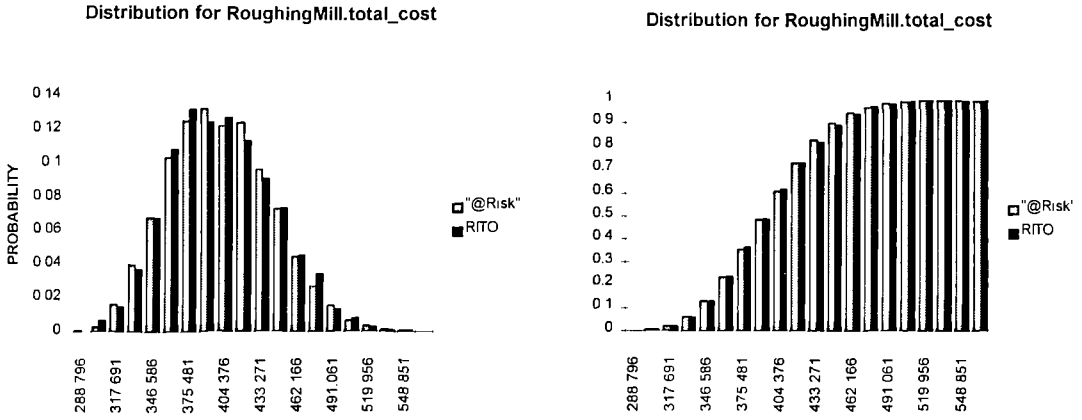


Figure 11-10: 5000 iterations

11.1.4.2 Convergence of Statistics and Simulation Time for RoughingMill.total_cost

In this section the convergence of the four most widely-used sample statistics are compared for samples of RoughingMill.total_cost generated by RiTo and by @Risk. Fifteen simulations were performed with each tool, consisting of 100, 200, 300,..., 1500 iterations. The mean, standard deviation, skewness and kurtosis (see [Freund 1980] for example for definitions) were calculated for each of the 30 sample data sets thus obtained. The results are presented in three forms. Firstly, as a table (Table 11-1). Secondly, as a plot of statistic against number of iterations (Figure 11-11 and Figure 11-12). Thirdly, as a plot of change in statistic against number of iterations (also Figure 11-11 and Figure 11-12).

For each tool the same seed was used for the random number generator for each of the ten simulations: a value of 1 was chosen.

#iterations	tool	mean	sd	skewness	kurtosis
100	@Risk	407.9917	43.2911	0.2511	2.9030
	RiTo	408.3310	43.4302	0.0757	2.3937
200	@Risk	408.0350	42.8353	0.2975	2.6576
	RiTo	408.0141	43.0841	0.2636	2.4124
300	@Risk	407.8610	40.8017	0.1676	2.7304
	RiTo	407.9748	40.9275	0.2304	2.5436
400	@Risk	407.7560	40.5419	0.1252	2.5951
	RiTo	407.9800	41.2825	0.2105	2.5103
500	@Risk	407.8087	41.4587	0.1675	2.5522
	RiTo	408.0573	41.2984	0.2405	2.6420
600	@Risk	407.9004	41.7320	0.2118	2.7444
	RiTo	408.0946	41.8643	0.2422	2.5786
700	@Risk	407.9542	41.7898	0.1949	2.8047
	RiTo	408.0594	42.4609	0.2675	2.5815
800	@Risk	407.9138	41.5168	0.1974	2.7970
	RiTo	408.0354	42.3812	0.2478	2.6057
900	@Risk	407.9036	41.4248	0.2144	2.8761
	RiTo	408.0045	42.0125	0.2579	2.6301
1000	@Risk	407.9169	41.5605	0.2413	2.8735
	RiTo	408.0017	41.8695	0.3046	2.6792
1100	@Risk	407.9340	41.7846	0.2596	2.8728
	RiTo	407.9898	41.8605	0.3045	2.7134
1200	@Risk	407.9747	41.6557	0.2788	2.8837
	RiTo	407.9925	41.6625	0.3075	2.7393
1300	@Risk	407.9810	41.6363	0.2789	2.8653
	RiTo	407.9698	41.3425	0.3137	2.7990
1400	@Risk	408.0005	41.5882	0.2830	2.9047
	RiTo	407.9800	41.6441	0.2953	2.7979
1500	@Risk	408.0135	41.6746	0.2853	2.9417
	RiTo	407.9414	41.6230	0.2800	2.7619

Table 11-1: Convergence of statistics for RoughingMill.total_cost

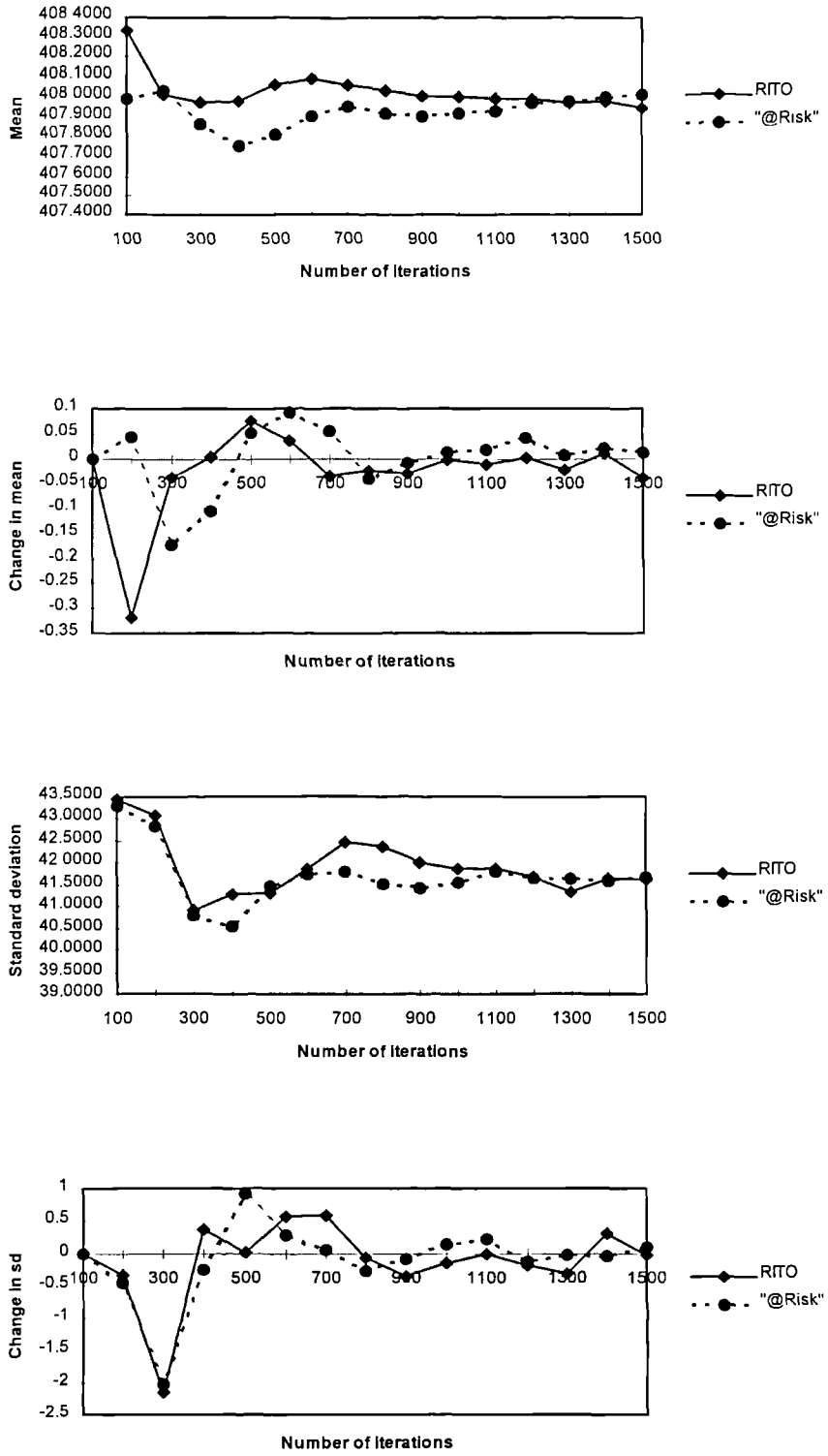


Figure 11-11: Convergence of mean and standard deviation for RoughingMill.total_cost

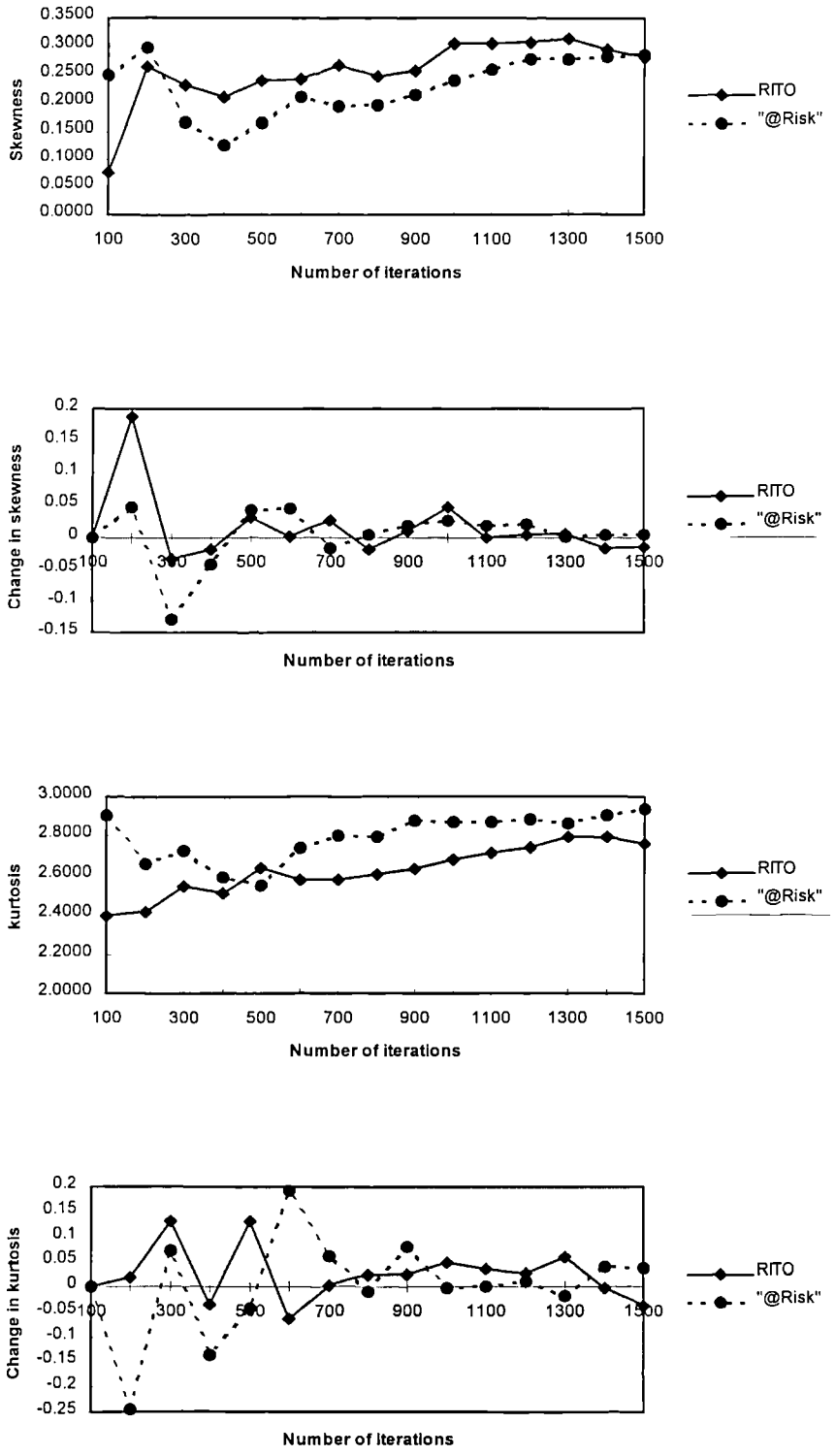


Figure 11-12: Convergence of skewness and kurtosis for RoughingMill.total_cost

Figure 11-13 shows a graph of calculation-time (in seconds) against number of iterations for @Risk and for RiTo. The measurements were taken on a PC with a 66MHz 486DX2 processor. As Figure 11-14 illustrates, RiTo is almost exactly 17 times faster than the @Risk simulation engine.

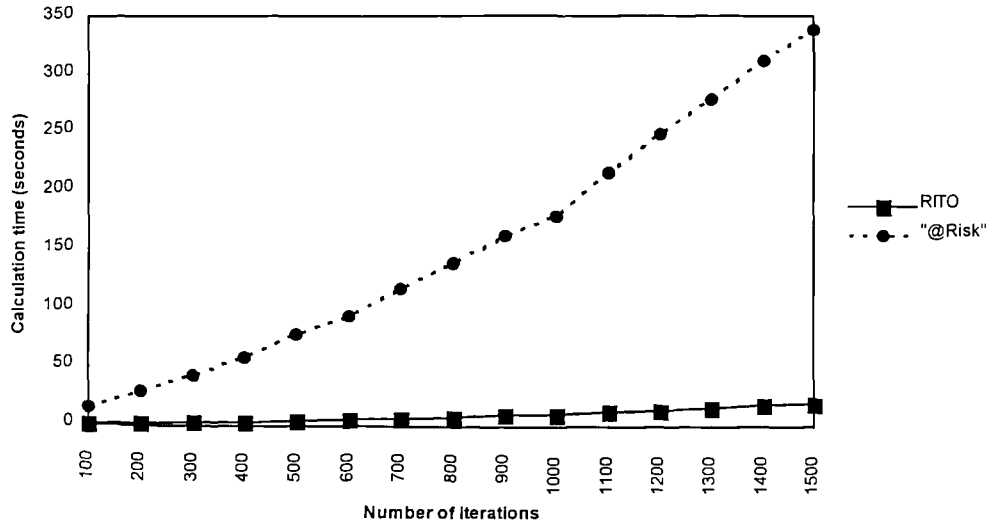


Figure 11-13: Calculation times to simulate RoughingMill.total_cost

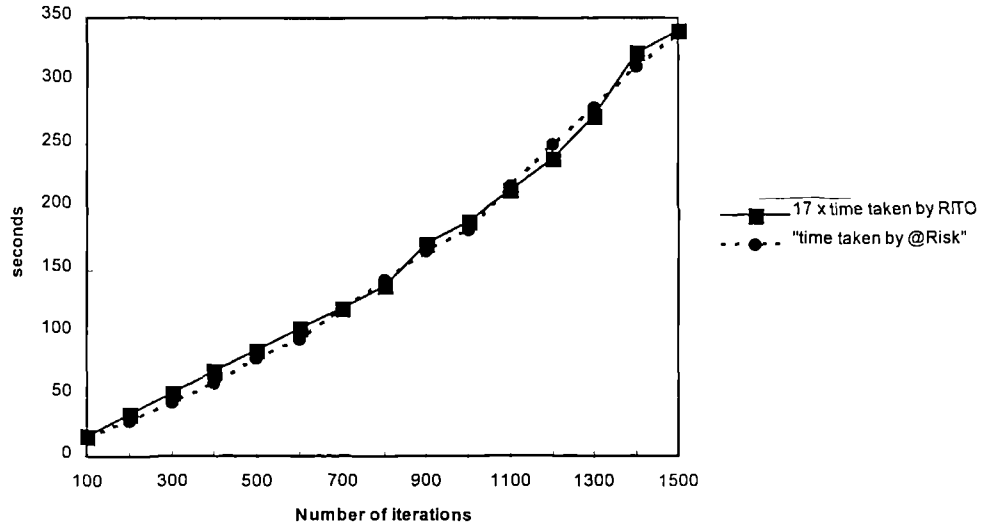


Figure 11-14: Normalised calculation times

11.1.4.3 Conclusions

The RiTo model has more generality than the spreadsheet model - for example, the RiTo model could be modified to represent design alternatives simply by the addition of objects. The RiTo model can also represent alternative derivation routes for values. The RiTo model aggregates six different types of cost independently through the hierarchy and provides a local cost total at each node, whereas the spreadsheet model only provided one dimension of cost. The RiTo model can be extended to include manufacturing processes, materials etc. for the physical design entities. Despite the greater generality of the RiTo model, and the simplicity of the case study, the spreadsheet model was considerably more complex and difficult to modify and maintain than the RiTo model.

The use of object classes in the RiTo model reduced the amount of redundant information stored. For example, in the spreadsheet model homogeneous and heterogeneous integration costs are stored for all "is-a-component-of" relationships, although they are only relevant to relationships between software components.

In the RiTo model this is expressed by defining a relationship class, derived from ICO ("is-a-component-of"), which may only be used to relate software components - this discipline can be imposed because of the existence of a class hierarchy. In the absence of a class hierarchy, the only way to ensure that the spreadsheet model is "safely" extensible is to give all objects all the data values which they might require.

The results obtained using Version 1.0 of RiTo agreed well with those obtained using the @Risk spreadsheet simulation engine. For 1500 iterations, the mean and standard deviation agree to 2 significant digits and the skewness and kurtosis agree to 1 significant digit. There was no significant difference between the two tools in the rate of convergence of sample statistics as the number of iterations is increased, and the convergence plots indicate that the two tools are converging on the same value for each statistic tested. RiTo is almost exactly 17 times faster than @Risk and there is no significant difference between the two tools in the non-linearity of calculation time as a function of number of iterations.

11.1.5 Object Instances in the Dynamic Positioning System Risk Model

The dynamic positioning (DP) system case study was modelled using the same set of classes as the roughing mill. Thus, building the risk model was very rapid. The simplified "Bill-of-materials" view of the DP model is shown in Figure 11-15.

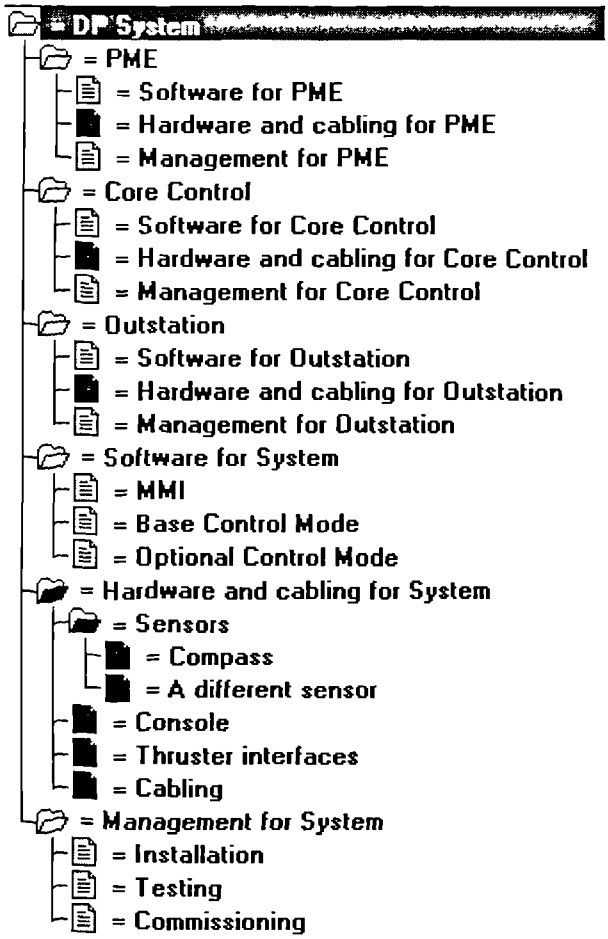


Figure 11-15: "Bill-of-materials" view of RiTo model for dynamic positioning (DP) system case study

There are several sources of uncertainty in the DP model:

- The number of Optional Control Modes is uncertain, as shown in Figure 11-16.
- The number of weeks of effort required for each Optional Control Mode is uncertain (represented by a uniform distribution).
- The number of weeks of effort required for the software and management tasks are uncertain (represented by a mixture of triangular and uniform distributions).
- A scenario is modelled where the design company is considering sub-contracting the software and management for the Outstation , but has not yet decided between two alternative contractors - further, for each possible contractor there is some uncertainty concerning what they will charge. Thus, the loaded_rates link for Outstation tasks, points to an IAO as shown in Figure 11-17, and hence to two alternative LoadedRates objects, each with uncertain attribute values.

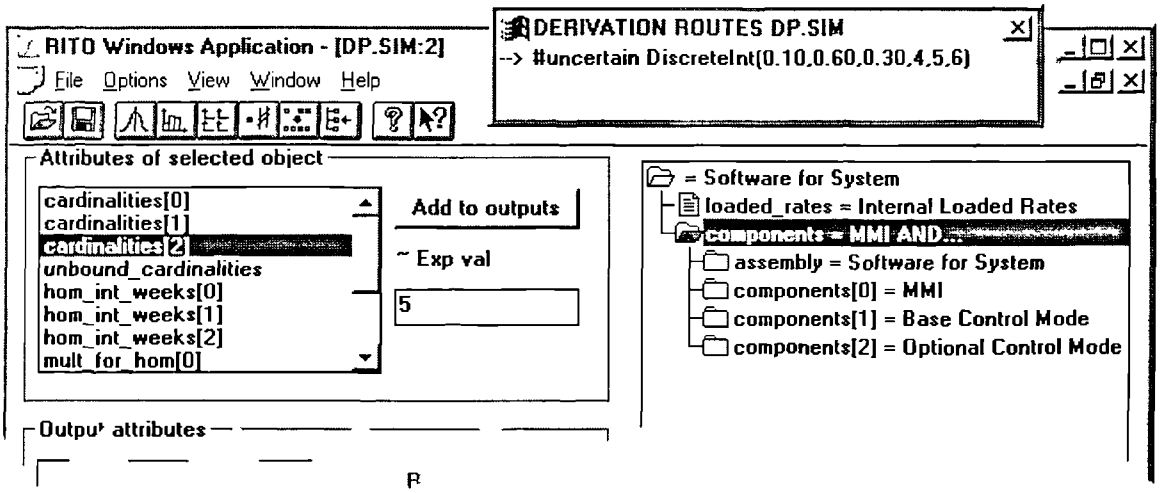


Figure 11-16: Uncertain number (4, 5 or 6) of Optional Control Modes for the DP case study

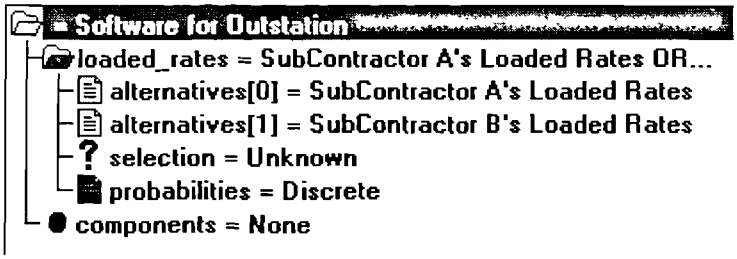


Figure 11-17: Alternative sub-contractors for the DP case study

There is only a very limited use of multiple derivation routes in the DP case study - they are used to ensure that if any CegSW or CegMan objects should subsequently be further decomposed into sub-tasks, then the costs obtained by aggregating the sub-tasks will be used in preference to the uncertain values currently given in the model. There is no use of heuristic rules in the DP risk model. The results obtained for the total cost distribution are shown in Figure 11-18, with a target value of £77K shown. To conclude, the control installation classes described in Section 11.1.1, together with the features automatically provided by RiTo, were found to be suitable for modelling the DP case study without further modification.

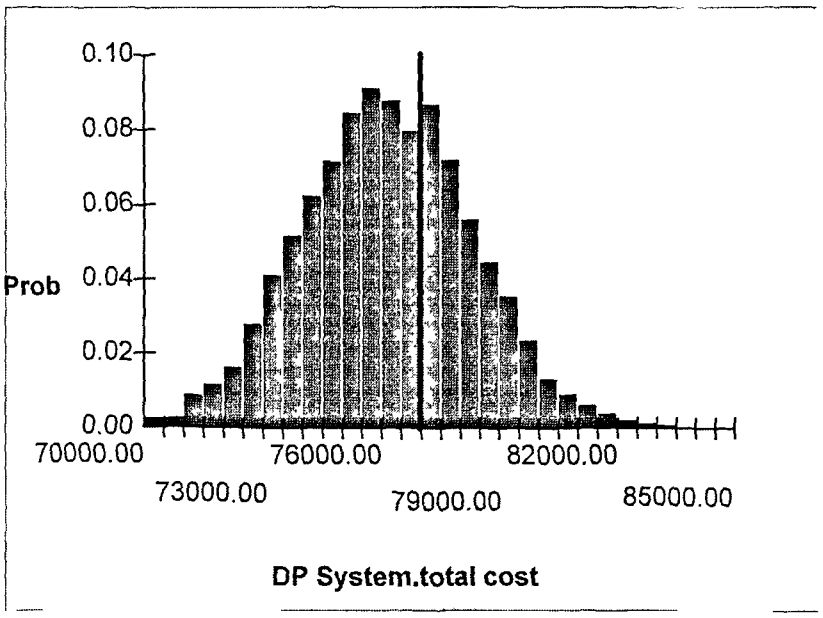


Figure 11-18: Example output from DP case study

11.2 Cost-Risk in Early Design of Interior Trim Area at Rover Group

11.2.1 Overview of Interior Trim Schema

The risk models described in this section were developed with Neil Davis of Warwick University in consultation with designers working in the Interior Trim area during a new vehicle development. Due to constraints of time, it was decided to define very few company-specific classes containing historical information and heuristic rules for this case study (although a couple of simplified examples were included as illustrations); instead the base classes (see Appendix D) were generally used as provided. Nonetheless, the case study provided a good test bed for most aspects of the methodology - particularly the product structuring model.

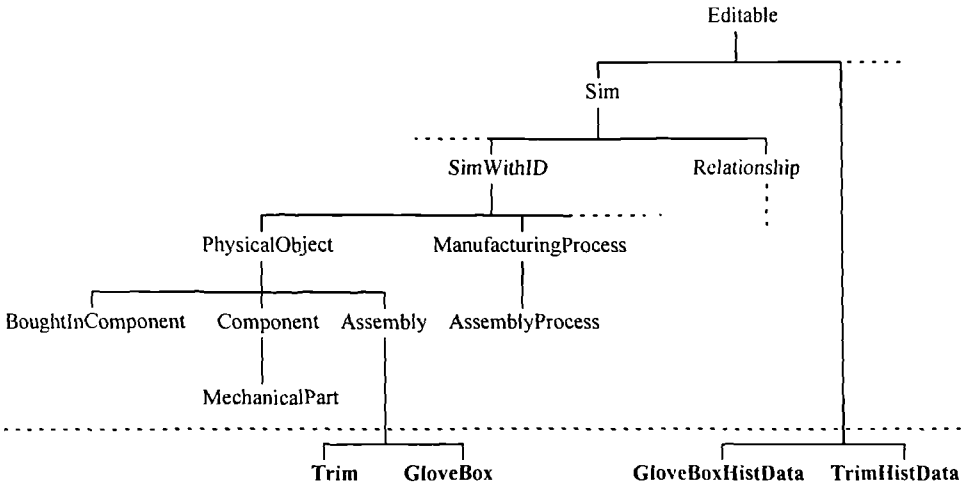


Figure 11-19: Partial inheritance hierarchy for base classes and exemplar trim area classes

Two exemplar Rover-specific classes were defined, namely `GloveBox` and `Trim` (see Figure 11-19 for the inheritance hierarchy and see Appendix H for a detailed description). Each of these classes was provided with a heuristic rule to calculate piece-part cost, using high-level attributes and historical data as input to the rule. The historical data was stored in objects of class `GloveBoxHistData` and `TrimHistData`

respectively. For both exemplar classes, the high-level attributes driving piece-part cost were the car size ("large", "medium" or "small") and the trim type ("soft feel", "hard feel" or "leather").

11.2.2 Object Instances in the Interior Trim Model

This section presents part of the risk model which was built for the Interior Trim case study, and provides numerical examples showing the types of information which such a model can provide to support early design decisions. In particular, examples are presented which use the direct calculation method for hierarchical risk sensitivity analysis (HRSA) described in Section 9.2.4.1 of Chapter 9. The current version of RiTo does not provide full and explicit support for HRSA; most of the algorithms described in Chapter 9 have been implemented, but they are incomplete and the user interface and graphics display has not yet been built. Therefore, the HRSA results presented here were generated using a spreadsheet to post-process the output obtained directly from RiTo.

Note: All numerical values have been changed to protect commercial confidentiality, and the object structures shown are incomplete and in some cases have been changed for illustrative purposes.

Figure 11-20 shows screen-shots from RiTo illustrating the BoM view of part of an early risk model of the interior trim area.

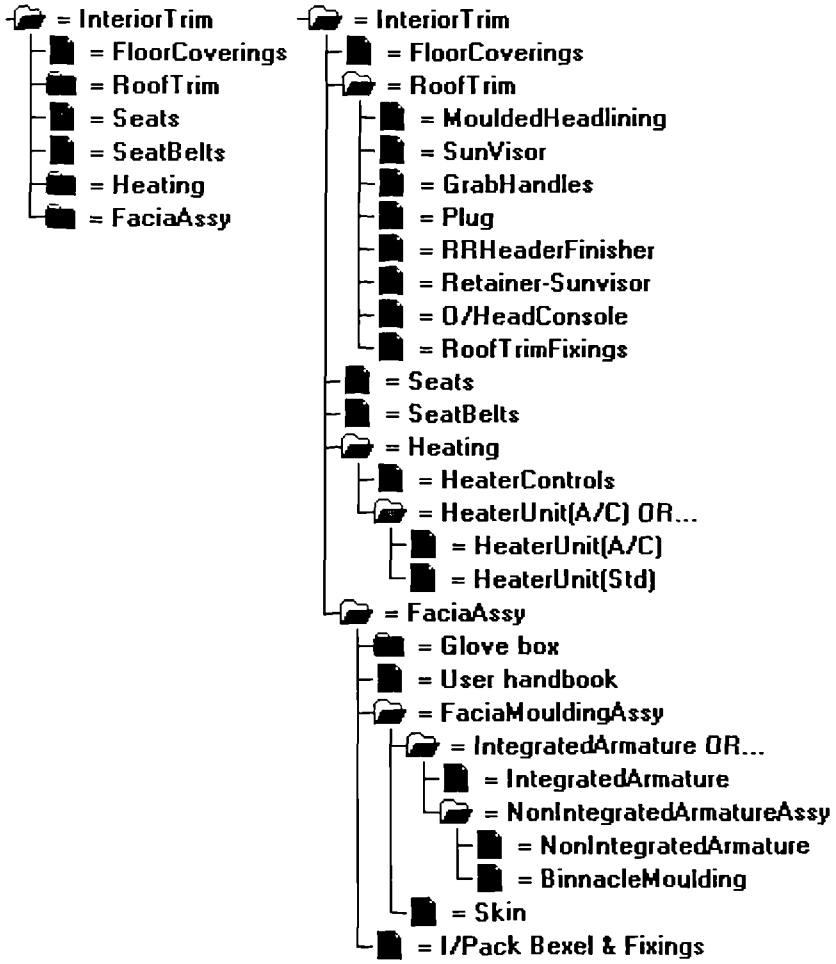


Figure 11-20: High level object structure of part of automotive risk model (Case 1)

At the time when this risk model was stored, there were two alternative designs for the heater unit under consideration, either the standard (D/C) unit or a novel design, which might for example have used an A/C power source. This has been modelled using an instance of the `RandomIAO` class which appears in the link tree as an object with name "HeaterUnit(A/C) OR..." (the suffix OR ... indicating that it is an `IAO`). In this situation, the engineer believed that there was a 60% likelihood that the A/C design would be adopted, and thus stored values of 40% and 60% into the `RandomIAO` object to reflect the probability of each alternative. There were also two alternative designs under consideration for the Armature in the Facia assembly, and these were modelled in a similar way.

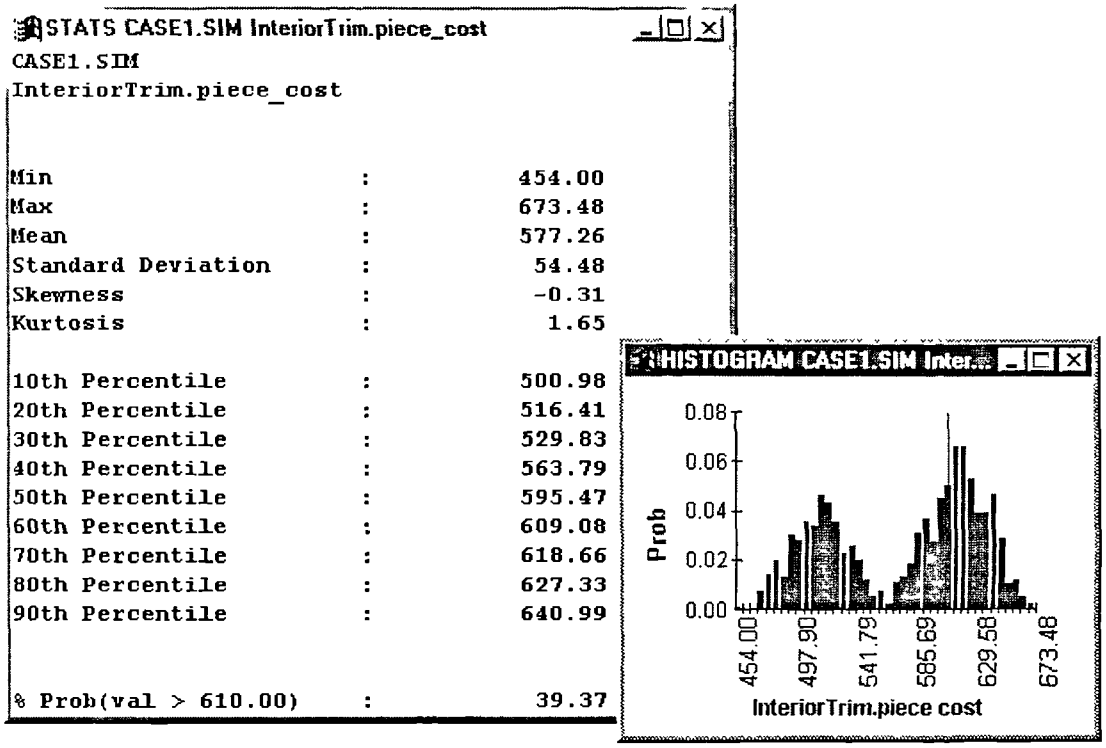


Figure 11-21: Risk model outputs at interior trim level, for piece cost (Case 1)

At this stage in the design process, the trim area team leader needs an indication of overall progress against budgets - is the project feasible? What is the probability of achieving budget? Figure 11-21 illustrates the types of graph and statistics RiTo can provide to address these questions, using piece cost as an example output attribute. A piece cost budget of £610 is shown (as a vertical bar on the histogram), and from the statistics box it can be seen that, based on the information which the design team have stored into the risk model, there is a currently a probability of approximately 60% that this budget will be achieved. The dual peak seen in the histogram is due to the two alternative heater unit designs under consideration . The 3σ value provides a standard measure of the overall level of piece cost risk for the interior trim area ($3 \times$ £54.48 $\approx$ £163, from Figure 11-21 and by monitoring this value at each design stage, the team leader can verify that the overall level of uncertainty is decreasing as the design proceeds and take corrective action if it is not.

Even at this early stage, the risk model contains fifteen independent continuous random variables, three discrete random variables (representing choices between alternatives) and numerous additional point-valued variables - and these are only the variables which contribute towards the interior trim piece cost, there are others which relate to tool and logistics costs. A version of the risk model from a later stage in the design

process contained over 500 objects and 93 continuous random variables contributing to interior trim piece cost alone. As design proceeds and the risk model becomes increasingly complex, reflecting the design complexity, it is clear that simple summary metrics such as those mentioned in the previous paragraph will become essential aids to the team leader's understanding of the overall risk levels in the vehicle area.

As explained in Chapter 9, in RiTo the problem of complexity is approached via hierarchical decomposition, and the BoM hierarchy is intrinsic to the HRSA approach to identifying critical uncertainties in the risk model. As the D-zero event approaches, when the overall feasibility of the vehicle programme will be judged, the trim area team leader must decide where best to concentrate resources so as to obtain the most accurate values possible for the costs and weight when the decision point is reached. The problem is to identify which are the major sources of the uncertainty in the output attribute - for example, interior trim piece cost. This is achieved using a measure termed the *risk sensitivity (RS)*. Recall that in Chapter 9, the risk sensitivity of an output attribute $\boldsymbol{Y}$ to an uncertain input $\boldsymbol{x}$ was defined as the expected value of the reduction in variance of $\boldsymbol{Y}$ when $\boldsymbol{x}$ is replaced by a point value.

Given a risk model containing many tens or hundreds of random variables, the team leader does not generally want (or need) to know to which individual random variables the overall cost risk is most sensitive. What is needed however, is a hierarchical breakdown of the sources of uncertainty, indicating for a particular node in the BoM hierarchy for example, which of its sub-assemblies make the major contribution to the uncertainty in the node's cost. The team leader may then choose to explore the most "risky" sub-assembly further, and so on. Thus the complexity of the sensitivity analysis is made manageable by decomposing it using the BoM structure; and the team leader obtains a high level overview of the "risky areas" where resources should be concentrated. In general, the inputs to the HRSA for a node (the sources of uncertainty to be ranked) will include both local attributes (belonging to the node), remote attributes (belonging to other nodes), and also both local and remote choices between alternative objects (the decision variables in `RandomIAOs`). In the following sections, several numerical examples of HRSA are presented which arose during the Interior Trim area case study. They are presented in order of complexity, beginning with the simplest case where the inputs are all independent and linearly related to the output and there are no choices between alternative objects involved.

11.2.3 HRSA Case 1: Piece costs for the interior trim area

The example shown in Figure 11-22 and Figure 11-23 was calculated numerically using the version of the risk model shown in the previous figures. This example corresponds to case (2) in Table 9-2 in Section 9.2.4.1 Chapter 9, where all the input variables are numerical, are interface attributes and are independent and there is a linear relationship with the output - thus the linear correlation coefficient is used to calculate the RS. Here it can be seen that the critical source of uncertainty in piece cost, at the interior trim level of the BoM, is the piece cost of the heating sub-assembly. And in this example it is clear that, were the team leader to choose to explore the heating sub-assembly further, he or she would then find that the critical source of uncertainty in its piece cost is the un-made design decision between the A/C or the standard heating unit (which gave rise to the dual peaks in the heating assembly's cost distribution shown in Figure 11-22). A virtue of the definition of risk sensitivity chosen in Chapter 9 is that it is equally applicable to numerical

random variables (such as costs) and to categorical random variables (such as the choice between alternative objects).

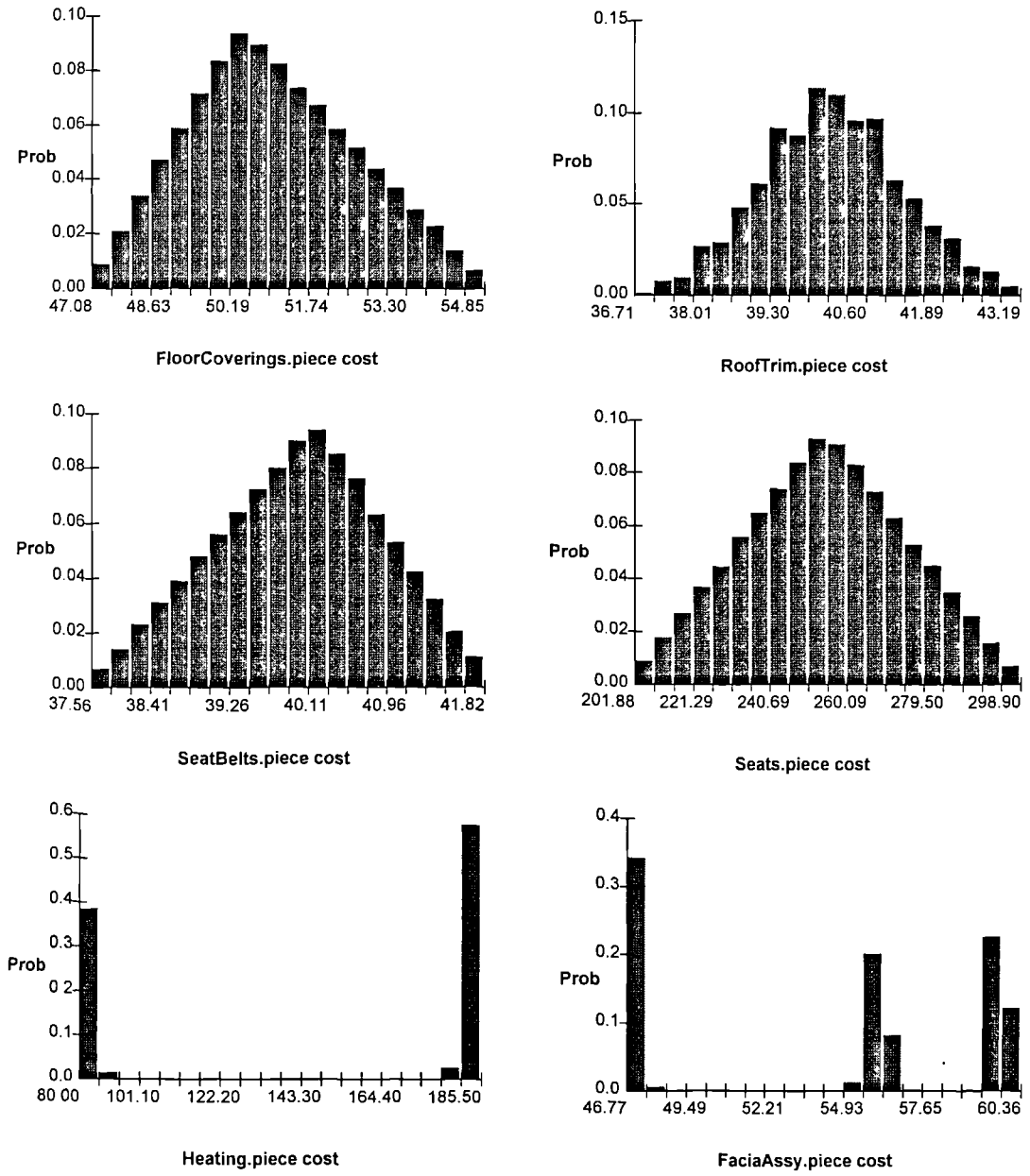


Figure 11-22: HRSA inputs for Case 1

This example is unusually simple in two respects. Firstly, the relationship between the output and the inputs is linear, with all coefficients being equal to 1 (the piece cost of the interior trim is simply the sum of the sub-assembly costs). Secondly, all the inputs are independent. Because of these two features, the RS value of each input can easily be calculated analytically (it is simply given by the input's variance), and also the RS values are additive - i.e. the variance in the output is equal to the sum of the RS values for the inputs. In Figure 11-23, the risk sensitivities have been expressed as standard deviations rather than variances - i.e. the square root of the calculated RS values have been plotted. Although this means that the RS values are no longer additive, this form has the advantage that the values depicted are commensurate with the spread (standard deviation) of the output attribute, which is a far more easily visualised statistic than the variance.

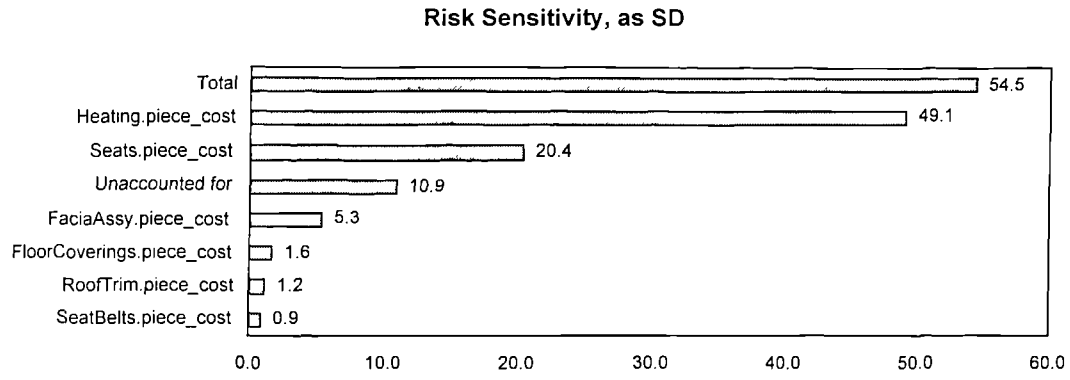


Figure 11-23: Results of HRSA for Case 1

RiTo must calculate the RS values numerically, using stored samples from previous Monte Carlo simulations[1] of the risk model and treating the methods which relate the variables as "black-boxes" (RiTo stores a Boolean flag indicating whether or not a method is linear, but the linear coefficients are unknown - the need to support arbitrary methods precluded more precise method characterisation). In the example shown, the RS value for each ***xi*** was evaluated by taking the square of the linear correlation coefficient of ***xi*** and ***Y*** samples, multiplied by the original variance of the ***Y*** sample (as specified in Chapter 9). The RS values thus obtained compared tolerably well with the variance values, as shown below, but sampling error lead to a 4% difference between the sum of the numerically evaluated RS values and the original variance of ***Y***. This difference, the output variance which is unaccounted for by the calculated RS values, is shown as a bar labelled "unaccounted for" in Figure 11-23.

```
InteriorTrim.piece_cost

InteriorTrim.piece_cost with Heating.piece_cost
Linear correlation                          :   0.923553
Variance                                    :   2408.11
Risk Sensitivity                            :                2531.22

InteriorTrim.piece_cost with Seats.piece_cost
Linear correlation                          :   0.416961
Variance                                    :   416.68
Risk Sensitivity                            :                 515.94

InteriorTrim.piece_cost with FaciaAssy.piece_cost
Linear correlation                          :   0.111750
Variance                                    :   27.89
Risk Sensitivity                            :                  37.06

InteriorTrim.piece_cost with SeatBelts.piece_cost
Linear correlation                          :   0.030396
Variance                                    :   0.81
Risk Sensitivity                            :                   2.74

InteriorTrim.piece_cost with FloorCoverings.piece_cost
Linear correlation                          :   0.022518
Variance                                    :   2.72
Risk Sensitivity                            :                   1.50

InteriorTrim.piece_cost with RoofTrim.piece_cost
Linear correlation                          :  -0.009552
Variance                                    :   1.38
Risk Sensitivity                            :                   0.27
                                                                    +
Sum of risk sensitivities                   :                3088.73

                                                             -3088.73
InteriorTrim.piece_cost Variance            :                2967.60
                                                                    +
Variance unaccounted for                    :                - 121.13
                                                             ( = - 4 )
```

[1] All Monte Carlo simulations used 1000 iterations in the case study examples and thus the sample size is 1000.

Taking as an example ***xi*** = FaciaAssembly.piece_cost, we can explicitly calculate the expected value of the variance reduction. The initial sample variance calculated for ***Y*** is 2967.60. If we then replace ***xi*** with a point value of X, we can evaluate the new variance of ***Y***. It is clear that the actual value of X selected has no effect on the variance of ***Y*** in this case, because the relationship is linear. Repeating this for a random selection of seed values we obtain the following results (recall that the RS for FaciaAssembly obtained using the correlation coefficients was 37.06).

seed	***xi*** = X	variance reduction in (***Y***)
0	55.5	105.5077
67	-	123.5914
82	-	97.38353
784	-	154.8688

Taking as another example ***xi*** = Heating.piece_cost, and recalling that the corresponding RS obtained using correlation was 2531.22, we obtain:

seed	***xi*** = X	variance reduction in (***Y***)
0	182	2522.496
67	-	2512.489
82	-	2509.048
784	-	2529.341

It is clear from these example results that, for a realistically sized model, using a realistic number of simulation runs (simulation time for 1000 runs on a 133 MHz Pentium with 32 MB RAM was approximately 4 seconds), the RS values calculated should not be interpreted as an exact measure of the expected variance reduction, but they do provide a useful indicative value and a ranking of sources of uncertainty.

Next, a selection of other examples from the case study is briefly presented. The examples do not all possess the two simplifying features of linearity and independence and they demonstrate the suitability of the numerical algorithms given in Chapter 9 for calculating an approximation to the risk sensitivities in more general circumstances.

11.2.4 HRSA Case 2: Piece cost of an injection moulding

This example is illustrated in Figure 11-24, and corresponds to case (1) in Table 9-2 in Section 9.2.4.1 Chapter 9, where the input variables are numerical, independent, are not interface attributes and taken individually, each is linearly related to the output - thus, once again, the linear correlation coefficient is used to calculate the RS values.

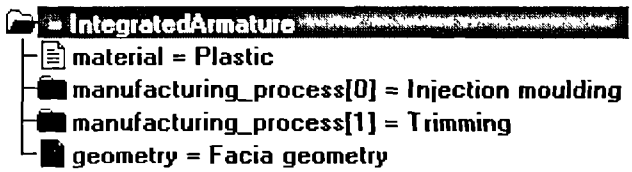


Figure 11-24: Object structure for an injection moulding (Case 2)

Here the piece cost for the armature is calculated from the product of the (uncertain) weight of the armature and the (also uncertain) unit cost of the material used (`Plastic`). The costs of the materials consumed by the manufacturing processes, and the cost of the processes themselves are also added. In this example there are three sources of uncertainty in the piece cost of the armature - the cost of the additional materials consumed by the injection moulding process is uncertain, as well as the weight and the unit cost. The attribute derivation network is given in Appendix J.

The distributions for the RSA inputs are shown graphically in Figure 11-25:

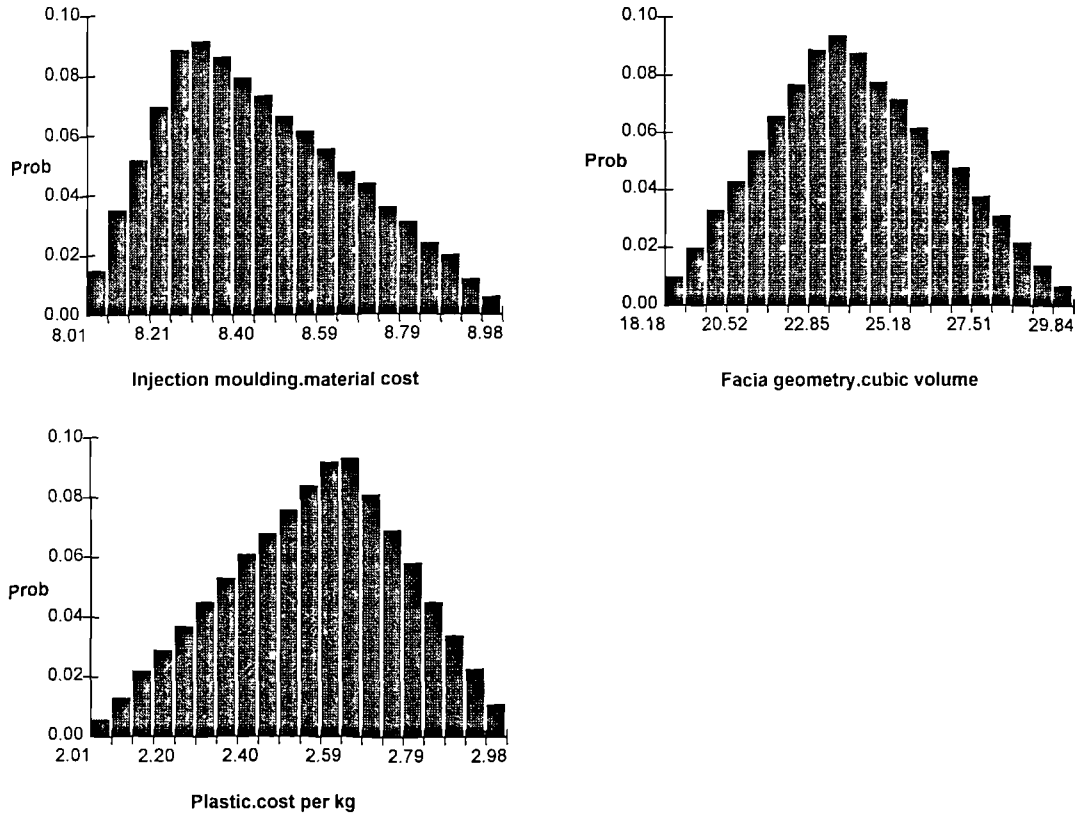


Figure 11-25: RSA inputs for Case 2

The results of the HRSA are as follows (and are also shown graphically in Figure 11-26):

```
IntegratedArmature.piece_cost

IntegratedArmature.piece_cost with Plastic.cost_per_kg
Linear correlation                                    :   0.581955
Risk Sensitivity                                      :                0.003297

IntegratedArmature.piece_cost with Injection moulding.material_cost
Linear correlation                                    :   0.201083
Risk Sensitivity                                      :                0.000394

IntegratedArmature.piece_cost with Facia geometry.cubic_volume
Linear correlation                                    :   0.760241
Risk Sensitivity                                      :                0.005627
                                                                                +
Sum of risk sensitivities                             :                0.009318

                                                                       -0.009318
IntegratedArmature.piece_cost Variance                :                0.009736
                                                                                +
Variance unaccounted for                              :                0.000418
                                                                       ( = 4%)
```

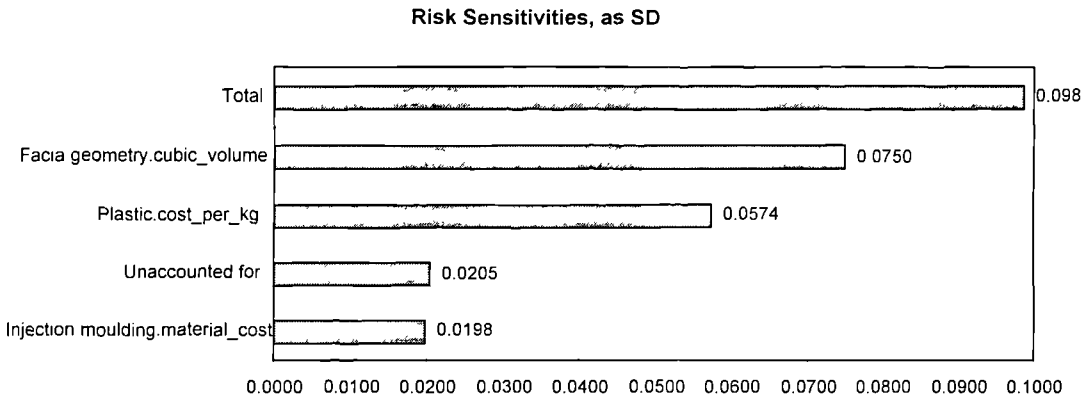


Figure 11-26: Results of Risk Sensitivity Analysis for Case 2

Using ***xi*** = Plastic.cost_per_kg as an example, we can now explicitly calculate the expected value of the variance reduction, and compare this with the RS value of `0.003297` obtained above. The value at which we choose to fix the cost_per_kg will now have an impact on the observed sample variance reduction in ***Y*** (unlike Case 1). We choose the 10th, 50th and 90th percentiles as the values at which to fix the cost_per_kg:

seed	***xi*** = *X*	Prob (***xi*** <= *X*)	variance reduction in ***Y***
0	2.244557	10	0.004723
67	-	-	0.004547
82	-	-	0.004712
784	-	-	0.004639
0	2.547570	50	0.003372
67	-	-	0.003172
82	-	-	0.003360
784	-	-	0.003277
0	2.799875	90	0.002115
67	-	-	0.001894
82	-	-	0.002101
784	-	-	0.002010

Thus we have approximated the continuous Triangular distribution of cost_per_kg by a distribution with three discrete values. The probabilities {*PX1, PX2, PX3*} which should be associated with each discrete value {*X1, X2, X3*} can be simply calculated using the method suggested in Chapter 9:

$$X1 = c^{-1}\left(\frac{PX1}{2}\right)$$

$$X2 = c^{-1}\left(PX1 + \frac{PX2}{2}\right)$$

$$X3 = c^{-1}\left(PX1 + PX2 + \frac{PX3}{2}\right)$$

where c is the cumulative PDF. Thus the probabilities are {20%, 60%, 20%) and using the results for a seed of 0.0, we obtain an expected value for the variance reduction of `0.00339`. The values obtained using the

other seed values are shown below and compare well with the numerically calculated RS value for `Plastic.cost_per_kg` of `0.003297`.

seed	variance reduction in *Y*
0	0.00339
67	0.003191
82	0.003379
784	0.003296

11.2.5 HRSA Case 3: Choice of assembly process depends upon component production volume

In Case 1, the relationship between the output and the inputs was linear and in Case 2 the output was still linearly related to each individual input. In this example, however the relationship is non-linear. Although there may well be parametric cost functions included in a risk model which are non-linear, the simplest example of a highly non-linear relationship is provided by a case-based method. A very general example of such a case-based method is the selection method of the `ByVolIAO` relationship class (see Appendix D), which selects one object from a set of objects on the basis of the production volume of a specified `PhysicalObject`. A variable cardinality attribute (an attribute which may take a variable number of values) called `ByVolIAO::prod_vols` defines the range of production volumes for which each alternative object will be chosen. Typically, this is used to select between different manufacturing processes for a component or assembly - since generally the choice of manufacturing process is heavily dependent on the estimated production volumes. An example is shown in Figure 11-27 and Figure 11-28 (see Appendix J for the attribute derivation network).

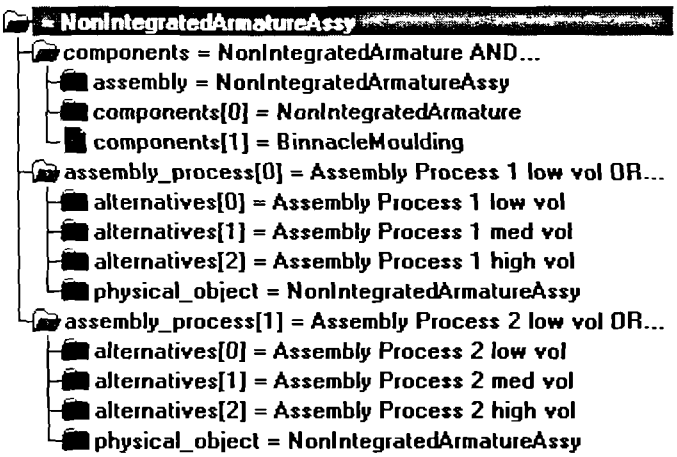


Figure 11-27: Containment hierarchy for Case 3

Here, the cost of assembling an armature assembly (`NonIntegratedArmatureAssy`) from its components is high for low production volumes, but for higher production volumes the use of more expensive manufacturing tools allows greater automation and hence lower assembly costs. The costs of the low and medium volume assembly processes are also themselves uncertain, and have different variances (the other assembly process costs are point valued). In this example, the uncertain piece costs of the components (`NonIntegrated Armature` and `Binnacle Moulding`) are unaffected by the production volume. The inputs to the HRSA are shown in Figure 11-28. Figure 11-29 illustrates the relationship between ***Y*** =

`NonIntegratedArmatureAssy.piece_cost` and each input (using scatter charts) - note in particular that the underlying relationship with production volume is a step function (i.e. not linear).

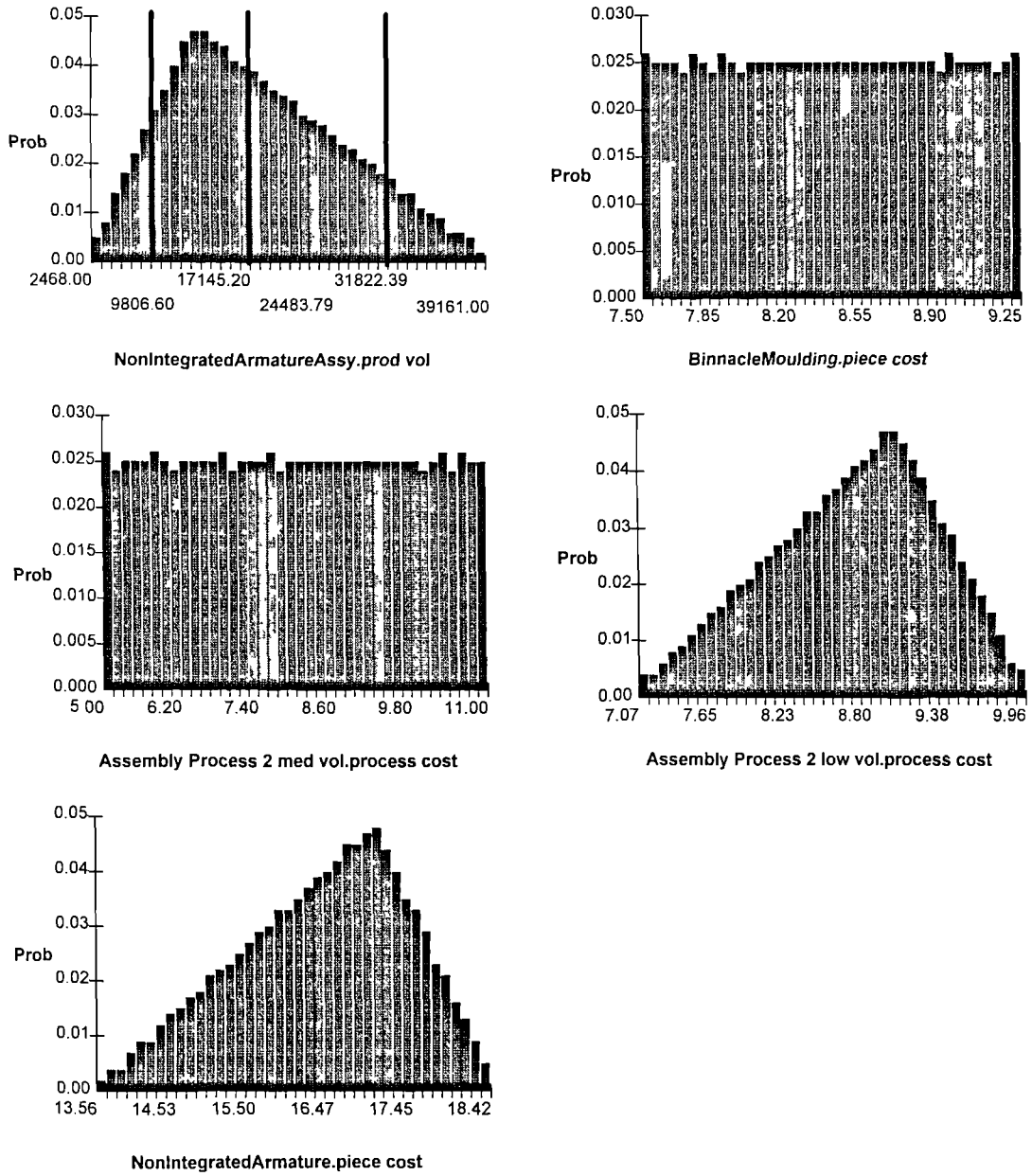


Figure 11-28: RSA inputs for Case 3

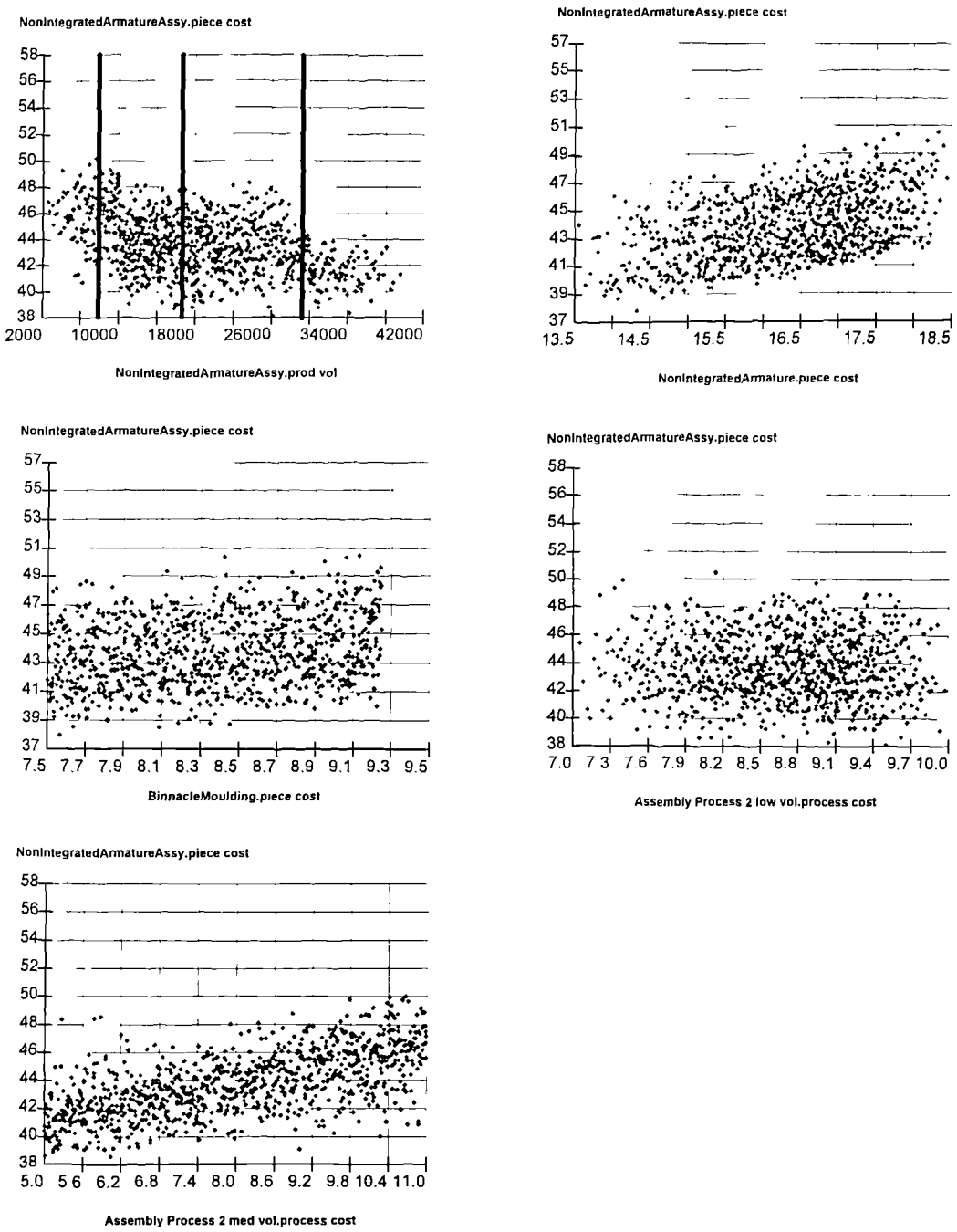


Figure 11-29: Scatter charts show relationships between Y and each input, for Case 3

A combination of three methods should be used in this case to numerically evaluate the risk sensitivity, according to the algorithm given in Chapter 9. For the RS to the piece costs of the two components, the linear correlation coefficient is used as previously, because the relationship is known to be linear (case (2) in Table 9.2 Chapter 9). The relationship with each uncertain process cost is also linear, but they are alternatives. Therefore, the correlation coefficient should only be calculated for the subset of the sampled values which were obtained using that particular alternative. Then the resultant variance reduction obtained should be multiplied by the probability of that particular alternative being selected (which can also be determined from the stored sample). The current version of RiTo cannot partition sampled values in this way, and thus RS values for the uncertain process costs must be calculated using the direct calculation method, ignoring the known linearity in the relationships.

And for the RS to production volume, the direct calculation method must also be used - where the random variable representing the production volume is approximated to a discrete rv and several sub-simulatations are performed. This corresponds to case (5) in Table 9-2 of Chapter 9. The three discrete values chosen for the discrete rv are the 10th, 50th and 90th percentiles as previously, and these are shown as vertical bars in Figure 11-28. The results obtained are:

production volume = *X*	Prob (*xi* <= *X*)	Variance of *Y*	variance reduction in *Y*
8138	10	4.268645	1.144809
16927	50	4.268645	1.144809
29653	90	1.296492	4.116962

which yields an expected value for the variance reduction of 1.7392392 and an expected value for the variance of 3.674215. The rest of the results are shown below and in Figure 11-30:

Assembly Process 2 low vol.process_cost = *X*	Prob (*xi* <= *X*)	Variance of *Y*	variance reduction in *Y*
7.771104	10	5.238372	0.175082
8.730634	50	5.339385	0.074069
9.451469	90	5.443645	-0.030191

which yields an expected value for the variance reduction of 0.0734196.

Assembly Process 2 med vol.process_cost = *X*	Prob (*xi* <= *X*)	Variance of *Y*	variance reduction in *Y*
5.597966	10	2.435625	2.977828
7.999684	50%	2.515272	2.898181
10.397916	90%	4.134093	1.279360

which yields an expected value for the variance reduction of 2.5903462. The overall results obtained are shown in Figure 11-30 and below:

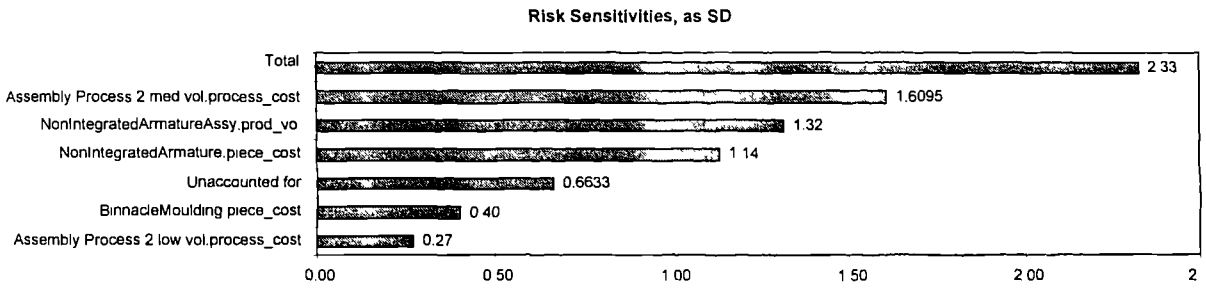


Figure 11-30: Results of Risk Sensitivity Analysis for Case 3

```
NonIntegratedArmatureAssy.piece_cost

NonIntegratedArmatureAssy.piece_cost with NonIntegratedArmatureAssy.prod_vol
Risk Sensitivity by direct calculation              :                    1.739239

NonIntegratedArmatureAssy.piece_cost with NonIntegratedArmature.piece_cost
Linear correlation                                  :  0.487927
Risk Sensitivity                                    :                    1.288796

NonIntegratedArmatureAssy.piece_cost with BinnacleMoulding.piece_cost
Linear correlation                                  :  0.172768
Risk Sensitivity                                    :                    0.161585


NonIntegratedArmatureAssy.piece_cost with Assembly Process 2 low vol.process_cost
Risk Sensitivity by direct calculation              :                    0.073420

NonIntegratedArmatureAssy.piece_cost with Assembly Pr cess 2 med vol.process_cost
Risk Sensitivity by direct calculation              :                    2.590346
                                                                                 +
Sum of risk sensitivities                                                5.853386

                                                                        -5.853386
NonIntegratedArmatureAssy.piece_cost Variance       :                    5.413454
                                                                                 +
Variance unaccounted for                            :                   -0.439932
                                                                         (= -8.1 )
```

In this particular example, because all the input distributions are known in closed form, and they are independent, it is fairly straightforward to find the expected value of the variance reduction obtained by fixing production volume using a (mostly) analytic method. The working is shown in Appendix I and the value calculated for the risk sensitivity to production volume is `1.68 +/- .02` which provides reasonable agreement with the value of `1.74` obtained using the numerical algorithm above. The RS values are no longer additive, now that the relationship is non linear, and hence the "variance unaccounted for" is relatively large (8% of the total variance).

If it were to be (incorrectly) assumed that the relationship with production volume is linear, then the value obtained for the RS to production volume would be `0.93`, which would lead to an incorrect ranking of the sources of uncertainty, with the production volume being ranked lower than the non-integrated armature piece cost. This example illustrates the need for direct calculation methods where non-linear relationships exist. Using rank correlation coefficients would also result in the same incorrect ranking:

Attribute	Rank Corr. Coeff.
Assembly Process 2 med vol.process_cost	0.647702
NonIntegratedArmature.piece_cost	0.459377
NonIntegratedArmatureAssy.prod_vol	0.422773
BinnacleMoulding.piece_cost	0.163972
Assembly Process 2 low vol.process_cost	0.004311

It can also be seen from this example that there is a danger, when approximating a continuous rv (such as production volume) to a discrete one that highly significant values of the discrete rv will be completely omitted. Examining the results shown above, it can be seen that none of the three production volumes tested lay within the intervals *I1* = [2000, 5000] or *I5* = (30000, 40000]. In this particular example, this was relatively unimportant for two reasons. Firstly, as shown in Table I-3 in Appendix I, the probability that the production volume will fall inside *I1* or *I5* is relatively small. Secondly, the variance in ***Y*** is not dramatically different if the production volume lies inside *I1* or *I5* than in the other cases (this can be seen from the scatter chart of production volume shown in Figure 11-29). In principle there is no upper limit to the potential

inaccuracy of this method, however equal spacing of the percentiles chosen as discrete values and also choosing a larger number of discrete values than 3 will reduce the likelihood of large errors. Comparison of the overall ranking with that obtained using rank correlation coefficients also provides an error check - if the rankings differ, the direct calculation method could be repeated with a larger number of discrete values.

11.2.6 HRSA Case 4: Piece cost of an injection moulding with alternative manufacturing processes

This example is similar to Case 2, except that at the design stage illustrated, there were three equi-probable alternative manufacturing processes under consideration (all injection moulding processes), with two of the alternatives having point values for their material costs and the third having an uncertain value. Thus far we have only considered numerical random variables as inputs to the risk sensitivity analysis. In this example however the risk sensitivity analysis includes both numerical variables and a categorical variable (the choice between alternative injection moulding processes). The example is shown in Figure 11-31 (see Appendix J for the attribute derivation network).

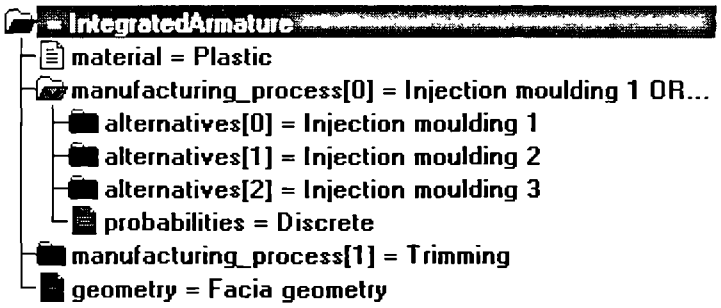


Figure 11-31: Object structure for alternative injection moulding processes (Case 4)

Four of the five inputs to the HRSA are shown in Figure 11-32 - the fifth input being the categorical random variable which selects between the three alternative injection moulding processes. The two vertical bars shown on the histogram for `injection moulding 2.material_cost` show the values of the two point-valued alternatives. It is not immediately obvious in this example which of these two sources of uncertainty makes the largest contribution to the overall variance - is it the material cost for Injection moulding 2 or is it the choice of alternative processes? If the point-valued alternatives were either very different from, or very similar to, each other and the mean for Injection moulding 2 then the answer would be intuitively clear. According to the algorithm given in Chapter 9, a combination of methods is used in this case to numerically evaluate the risk sensitivity. For the RS to `Plastic.specific_gravity`, `Plastic.cost_per_kg` and `Facia geometry.cubic_volume`, the linear correlation coefficient is used as previously, because the relationship between the output and each of these inputs taken individually is known to be linear (this is case (1) from Table 9-2). For the RS to `Injection moulding 2.material_cost` however, the underlying relationship can no longer be regarded as linear because of the existence of alternative processes, and thus the direct calculation method must be used, where the (continuous) material cost rv is approximated to a discrete rv (case (3) in Table 9-2). For the RS to the process selection variable, a direct calculation method should also be used (case (4) from Table 9-2), where the sample of $\boldsymbol{Y}$ values is partitioned according to the value of the selection variable, and the variance of each partition (each sub-sample) is then evaluated. The current version of RiTo does not explicitly support this

algorithm, however very similar results were be obtained by fixing the process selection variable with a value of process #i and then performing a smaller simulation with 1000 × probability (process #i) runs and calculating the variance of the output. This was repeated for each of the three values of i. The only difference between this approach and partitioning an existing stored sample for ***Y*** is that the benefits of Latin Hypercube sampling will be less apparent when an existing sample is partitioned - and thus more iterations may be required to achieve equally accurate results. The results obtained are:

xi = RandomIAO.selection = *X*	Prob(*xi* = *X*)	Number of runs	variance reduction in ***Y***
Injection moulding 1	0.333333	333	0.037444
Injection moulding 2	0.333333	333	-0.019478
Injection moulding 3	0.333333	334	0.037115

Which yields an expected value for the variance reduction of 0.003225.

Injection moulding2.material_cost = *X*	Prob (*xi* <= *X*)	var. reduction in ***Y***
7.346732	10	0.007062
7.590574	50	0.016831
7.971260	90	-0.020803

Which yields an expected value for the variance reduction of 0.007350. The overall results are shown below and in Figure 11-33, where it can be seen that, of the two, it is the material cost for Injection moulding 2 which makes the largest contribution.

```
IntegratedArmature.piece_cost

IntegratedArmature.piece_cost with Plastic.specific_gravity
Linear correlation                                    :   0.038643
Risk Sensitivity                                      :              0.000097

IntegratedArmature.piece_cost with Plastic.cost_per_kg
Linear correlation                                    :   0.386911
Risk Sensitivity                                      :              0.009729

IntegratedArmature.piece_cost with Injection moulding 2.material_cost
Risk Sensitivity                                      :              0.007350

IntegratedArmature.piece_cost with Facia geometry.cubic_volume
Linear correlation                                    :   0.516738
Risk Sensitivity                                      :              0.017353

IntegratedArmature.piece_cost with RandomIAO.selection
Risk Sensitivity                                      :              0.003225

Sum of risk sensitivities                             :              0.037754

IntegratedArmature.piece_cost Variance                :              0.064989
```

The same ranking of numerical variables is obtained using the rank correlation coefficient:

Attribute	**Rank Corr. Coeff.**
Facia geometry.cubic_volume	0.480995
Plastic.cost_per_kg	0.326391
Injection moulding 2.material_cost	0.298853
Plastic.specific_gravity	0.033220

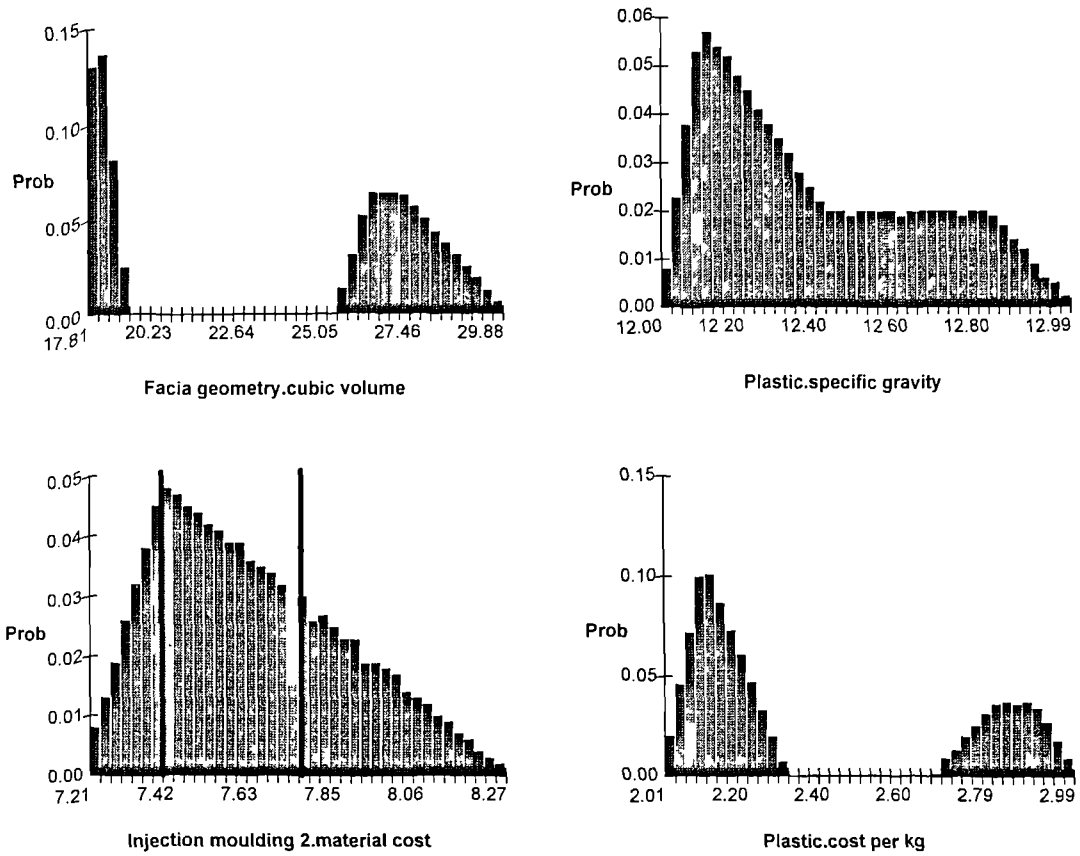


Figure 11-32: Numerical HRSA inputs for Case 4

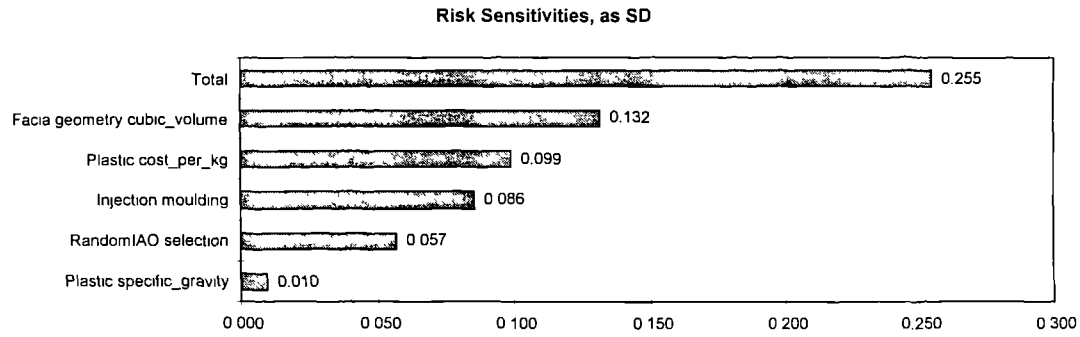


Figure 11-33: Results of RSA for Case 4

In conclusion, it can be seen from this example that risk sensitivity, as defined in Chapter 9, provides a meaningful measure of contribution towards uncertainty which can be applied to both categorical and numerical variables. The risk sensitivities cease to be additive where there are alternative objects, and hence the "variance unaccounted for" can no longer be used as an indication of the validity of the analysis (in this example it takes a value of 42%). However, a comparison of the ranking of numerical variables obtained with that given by the rank correlation coefficient can still be performed and will still be of value.

11.2.7 HRSA Case 5: Introduction of Variant Assemblies

In the final example, the risk model had been refined to include some variant parts (see Chapter 9, Section 9.3). The example also shows how a HRSA can be performed on a risk model which includes several

different product models (variants within a vehicle program), with a different estimated sales volume for each product but with many components and assemblies shared between products. Three product models (`Product A`, `Product B` and `Product C`) were introduced into the risk model and Figure 11-34 shows part of the component model (the BoM tree view) when `Product A` is selected.

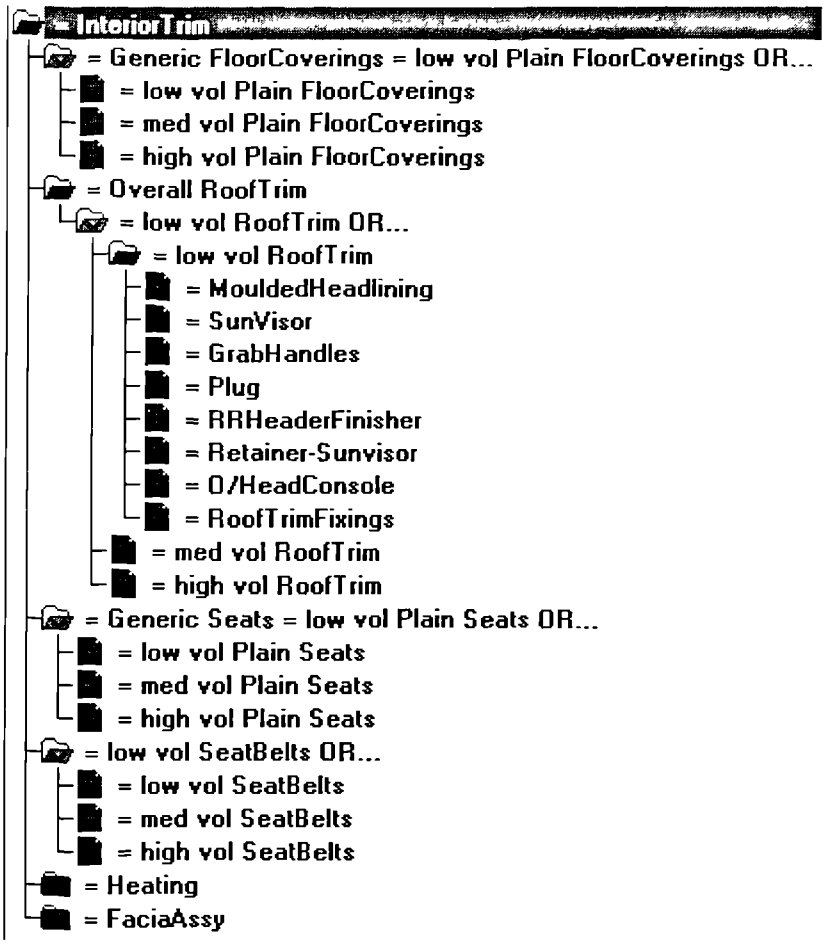


Figure 11-34: BoM view of object structure for Case 5, with product A selected

Using RiTo, the user may select the product of interest and the BoM tree is automatically "pruned", to display the component configuration for the selected product. The `IVO` relationships are not therefore explicitly shown in the BoM tree view (the more detailed link view must be used to explore them) but in Figure 11-34 the existence of variants can be deduced from lines of the form "Generic <generic object name> = <specific object name>" such as:

```
Generic FloorCoverings = low vol Plain FloorCoverings OR...
```

This indicates that the plain variant has been selected for the floor coverings and placed in the BoM tree in the position occupied by the `Generic FloorCoverings` object. But, there are actually three alternative plain floor covering objects stored in the model, each corresponding to a different production volume, so it is in fact the set of three alternative plain objects which has been selected, and this is the significance of the `OR...` appended to the end of the line. The attribute derivation network is shown in Appendix J.

In this example, all three product models have an uncertain sales volume (all modelled as triangular distributions). `Product A` is the basic model, `Product B` has a sporty trim and `Product C` has both a

sporty trim and driver and passenger airbags (Figure 11-35). The production volumes for the components and sub-assemblies in the `InteriorTrim` assembly have been calculated automatically by RiTo from the uncertain sales volumes - some assemblies are required for all 3 products, and some are only required for certain products. Note that the `Interior Trim` assembly has six sub-assemblies - `FloorCoverings`, `RoofTrim`, `Seats`, `SeatBelts`, `Heating` and `FaciaAssy`. The `FloorCoverings` and the `Seats` have a sporty variant and a plain variant - which variant is included in the BoM depends upon the selected product. In the `FaciaAssy` there is an `Airbag` assembly which is only included in the BoM when `Product C` is selected. Also in the `FaciaAssy`, there are four variants of the `SteeringWheel` assembly, and again which variant is included in the BoM depends upon the selected product.

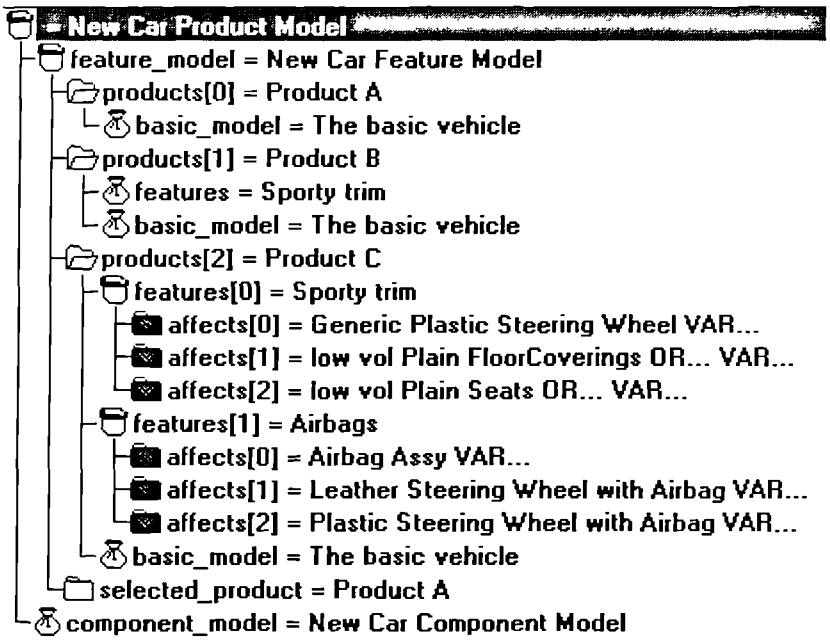


Figure 11-35: Part of feature model for Case 5

For each of the sub-assemblies `FloorCoverings`, `RoofTrim`, `Seats` and `SeatBelts` (and each variant where they exist) there are three alternatives and the choice between these alternatives is determined by the production volume for the sub-assembly. In this example, therefore, the inputs to HRSA are no longer independent:

Attribute	Depends upon
vA = Product A.sales_volume	
vB = Product B.sales_volume	
vC = Product C.sales_volume	
c1 = Generic FloorCoverings.piece_cost	vA
c2 = Overall RoofTrim.piece_cost	vA, vB, vC
c3 = Generic Seats.piece_cost	vA
c4 = low vol SeatBelts.piece_cost	
c5 = Heating.piece_cost	
c6 = FaciaAssy.piece_cost	vA, vB, vC

Table 2: Inputs to RSA for Case 5, with product A selected

The results of the HRSA will depend upon which of the three products have been selected. In this example, we perform the HRSA on `Product A`. The inputs to the HRSA for `Product A` of the `InteriorTrim` container are shown in Table 2. Note that all three sales volumes have an effect on the total piece part cost for all three products - because there are some assemblies (`RoofTrim`, `SeatBelts`, `Heating`, `FaciaAssy`) which are used in all three products and thus their production volumes depend upon the sum of the three product sales volumes.

The method for calculating the RS to the sales volumes is, as previously, direct calculation since they are independent rvs with a non-linear relationship to the output, Y = `InteriorTrim.piece_cost`. As usual, we approximate each sales volume to a discrete rv with three values and perform three sub-simulations. The cost of the heating assembly is unaffected by sales volumes and is known to be linearly related to Y, thus its RS value is calculated using the linear correlation coefficient, as in previous examples. The low volume seat belts assembly is also known to be unaffected by sales volumes and the underlying relationship with Y is linear, but because it is one of three alternatives, the linear correlation coefficient can no longer be used. As with the alternative process costs introduced in case 3, the proposed method is to partition the sample, but since this is not currently supported by RiTo, the direct calculation method is used instead. The RS to the costs of the floor coverings and the seats are calculated using the partial correlation coefficients $\rho_{Yc1\,vA}$ and $\rho_{Yc3\,vA}$, which remove the effect of vA on the variance reduction. The RS to cost of the facia assembly and roof trim are calculated using the partial correlation coefficients $\rho_{Yc6\,vAvBvC}$ and $\rho_{Yc2,vAvBvC}$, which remove the effect of vA, vB and vC on the variance reduction The results are shown below and in Figure 11-36.

ProductA.sales_volume = X	Prob (xi <= X)	var. reduction in Y
2560	10	1545.116156
5124	50	1946.125379
7818	90	1926.806906

ProductB.sales_volume = X	Prob (xi <= X)	var. reduction in Y
8663	10	-69.77903697
18831	50%	157.9853428
27786	90%	376.0949948

ProductC.sales_volume = X	Prob (xi <= X)	var. reduction in Y
3793	10%	2.051030143
7884	50%	- 68.92020123
10476	90%	-195.6099847

low vol SeatBelts.piece_cost = X	Prob (xi <= X)	var. reduction in Y
38.548299	10%	25.61224748
39.843673	50%	13.27014897
40.983000	90%	1.94039517

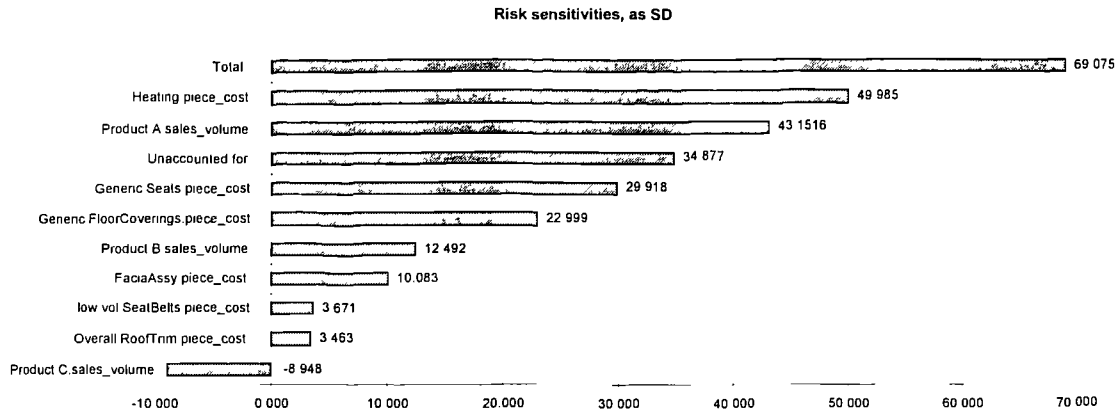


Figure 11-36: Results of HRSA for Case 5

```
InteriorTrim.piece_cost

InteriorTrim.piece_cost with Product A.sales_volume
Risk sensitivity                                        :              1862.059840
[Risk sensitivity calculated linearly                   : 1533.407256]

InteriorTrim.piece_cost with Product B.sales_volume
Risk sensitivity                                        :               156.0543972

InteriorTrim.piece_cost with Product C.sales_volume
Risk sensitivity                                        :              -80.06391165

InteriorTrim.piece_cost with Generic FloorCoverings.piece_cost
(excluding ProductA.sales_volume)
Partial correlation                                     : 0.332950
[Linear correlation                                     : 0.629706]
Risk sensitivity                                        :               528.931673

InteriorTrim.piece_cost with Overall RoofTrim.piece_cost
(excluding ProductA.sales_volume,ProductB.sales_volume, ProductC.sales_volume)
Partial correlation                                     : 0.050130
[Linear correlation                                     : 0.185437]
Risk sensitivity                                        :                11.990490

InteriorTrim.piece_cost with Generic Seats.piece_cost
 excluding ProductA.sales_volume)
Partial correlation                                     : 0.433129
[Linear correlation                                     : 0.666300]
Risk sensitivity                                        :               895.109283

InteriorTrim.piece_cost with low vol SeatBelts.piece_cost
Risk sensitivity                                        :              13.47261791

InteriorTrim.piece_cost with Heating.piece_cost
Linear correlation                                      : 0.723639
Risk sensitivity                                        :              2498.535160

InteriorTrim.piece_cost with FaciaAssy.piece_cost
(excluding ProductA.sales_volume,ProductB.sales_volume, ProductC.sales_volume)
Partial correlation                                     : 0.145973
[Linear correlation                                     : 0.149512]
Risk sensitivity                                        :               101.668544
                                                                                  +
Sum of risk sensitivities                               :              5987.758093

                                                                      -5987.758093
Interior Trim.piece_cost Variance                       :              4771.352862
                                                                                  +
Variance unaccounted for                                :             -1216.405231
                                                                       ( = -25 )
```

The variance unaccounted for is approximately -25% of the total variance; but we would not expect the risk sensitivities to be additive, since the relationship between Y and the sales volumes is not linear and the partial correlation coefficient only removes that part of the variance reduction which can be explained by a linear regression.

The risk sensitivity to `ProductC.sales_volume` is negative - this suggests that fixing the sales volume of product C could be expected to increase the variance in *Y*. However, the variance reduction is approximately 1.7% of the overall variance and this is less than the sampling error. The sampling error is estimated by performing ten simulations with different seed values and calculating the three-sigma value for the variance of *Y*, which is `724.3747` or approximately 15% of the overall variance. This suggests that the RS values for `Seats`, `Heating` and `ProductA.sales_volume` are reliable, but the other values should be treated with caution. Once again, however, the algorithm has successfully identified the critical uncertainties in the risk model.

Notice that the linear correlation with `FaciaAssembly` cost is very similar to the partial correlation - because, as mentioned earlier, the effect of the sales volumes on the `FaciaAssembly` is negligibly small. Whereas, the linear correlation with the cost of `Seats` is considerably larger than the partial correlation coefficient, illustrating the influence of the sales volume.

Amongst the sales volumes, in this case we find (unsurprisingly) that the uncertainty *in* the sales volume for `ProductA` has the major effect on the uncertainty in the piece part cost for `ProductA`. However, if we repeat the RSA with `ProductC` selected we discover that it is in fact the sales volume for `ProductB` which makes the main contribution to the cost uncertainty for `ProductC`. This is clearly possible, since products B and C (both having a sporty trim) share many components and `Product B` has a larger variance in cost than `Product C`. This illustrates the kind of, perhaps non-obvious, results which can be obtained using a combined feature and component model which incorporates uncertainty.

11.3 Summary and Concluding Remarks

This chapter presented risk models which have been built for each of the case studies first introduced in Chapter 7. The classes used in these models were described, and the models themselves presented with some evaluation results.

The risk model built for the steel roughing mill case study was used as the basis for a comparison between RiTo and a commercial Monte Carlo simulation package. This served as a testing and validation exercise for RiTo. No significant differences were observed in the numerical results obtained; the expected values agreed to two decimal places and the probability distribution histograms obtained were compared visually and found to agree. Graphs were presented showing how various statistics of the output sample (plotted on the y- axis) varied against number of simulation runs (plotted on the x-axis). The statistics tested were the mean, standard deviation, skewness and kurtosis. These graphs showed no significant difference in the values to which the statistics converged or the rates of convergence between RiTo and the commercial package. A comparison of simulation time found that RiTo was approximately 17 times faster than the commercial package.

The risk model of the dynamic positioning system, built from the same set of classes as the steel roughing mill model, was described and some example outputs were presented.

The interior trim risk model was built almost entirely from the base classes - however, some exemplar Rover-specific classes were introduced, to illustrate how historical information can be re-used. The "direct

calculation" method for hierarchical risk sensitivity analysis (HRSA) presented in Section 9.2 of Chapter 9 was applied to several examples taken from this case study, one of which included variant parts, and both the working and the results obtained were presented here. Some of the HRSA results were compared with analytically calculated values, or with other numerically calculated values. It is concluded that useful but approximate results can be obtained using the numerical algorithms for HRSA which have been developed. As a means of identifying the critical uncertainties, the algorithms were found to be adequate. However, in some circumstances the estimate of RS obtained will be inaccurate; it is important that possible sources of inaccuracy should be identified and flagged to the user (for example insufficient number of simulation runs, the presence of highly non-linear and discontinuous methods or the use of a linearity assumption to remove the influence of an independent variable on a dependent variable). Results for numerical attributes should also be compared with the results obtained using the rank correlation coefficient, as proposed in Section 9.2 of Chapter 9, and if the ordering obtained differs, the user should be warned.

The case studies presented did not fully exploit all aspects of the RiTo tool and methodology; in particular, few meaningful company-specific inherited classes were developed. Thus some potential benefits of the O-O approach - re-use of historical information and of estimation methods via object classes - were not realised. However, the Rover case study showed that this "base-class-only" mode of use was adequate and should perhaps be regarded as an initial use mode which can be implemented rapidly, with the encapsulation of estimation methods and historical data into more specialised object classes being undertaken subsequently in a more detailed level of implementation. An important conclusion which was reached early in the case study, however, was that variant forms for components and assemblies are considered even in early design and hence the combined feature and component model described, which allows both variants and design alternatives to be modelled, was both found to be necessary and shown to be adequate.

The Rover case study also omitted explicit modelling of "risks" and "opportunities" (see Chapter 7) as lists of discrete, uncertain, cost variations (accompanied by descriptive text) stored with each object. The methodology easily supports this approach - this would provide numerical support for the risk register advocated in traditional project risk management [Carter 1994], with class-specific default values for the lists providing a risk checklist for the designer as the risk model is built. Strictly speaking, the causes of the cost variations should be included in the risk model and the ability to add arbitrary lists of variations is a move away from a purely O-O approach, however risk lists provide a simple way to compensate for the finite size of any practical model.

The case studies indicated the need for simple summary metrics to provide an overview and illustrated how "3 sigma" values, the probability of achieving budget and the expected value of the variance reduction (HRSA) can help satisfy this need. The traditional project risk management approach to this problem is purely qualitative - for example, providing the project manager with a count of the number of "high", "medium" and "low" risks present in a project (perhaps presented graphically to indicate their distribution over the project). It is suggested that the methodology presented in this thesis provides information which is more directly relevant to the decisions faced, and that the cost of gathering this information is much lower than in traditional spread-sheet based risk models.

11.4 References

[Crossland 1996] Crossland, R., "Evaluation of Version 1.0 Risk Tool", *STARTED report STARTED/BRI/013/1*, January 1996.

[Freund 1980] Freund, J. E. and Walpole, R. E., "*Mathematical Statistics*", Prentice Hall International Editions:London, 1980.

CHAPTER 12

Conclusions and Further Work

This final chapter summarises the work reported in the thesis, and its limitations, and suggests directions for future research to develop the risk modelling methodology and tool which have been presented. In the first section, the conclusions which can be drawn from the research are summarised. The findings concerning the nature of uncertainty in early design (and the process by which the consequent risk is managed) are outlined; the main features of the modelling methodology and tool which have been developed are presented and conclusions are drawn regarding their suitability for modelling the case studies with the industrial partners; finally the risk model evaluation methods which have been developed for risk sensitivity analysis and for the modelling of design variants are summarised. The second section details the limitations of the case-studies reported in the thesis, and also presents the limitations of the risk modelling methodology and tool. In the third section, some areas of future work to extend the usefulness of the methodology and tool are suggested and the final section summarises the bearing of the research upon the initially stated problem of supporting design engineers and their managers in decision-making during the early stages of the design process - particularly in bidding situations.

12.1 Summary of Conclusions From the Research

12.1.1 What is a Risk Model

In this thesis, a *risk* was defined to be any uncertain event which may have a detrimental impact on the outcome of the design project. Conversely, an *opportunity* was defined to be any uncertain event which may have a desirable impact on the outcome of the design project. The *level of risk* was defined to be a combination of the probability of occurrence of an uncertain and undesirable event and the impact of that event. And the *level of opportunity* was defined to be a combination of the probability and impact of a desirable uncertain event. A *risk model* represents both risks and opportunities and can be used to evaluate levels of both risk and opportunity.

12.1.2 Requirements for a Risk Model

Having defined these terms, the requirements for a risk model for use in early design were explored. The nature of the design process was investigated by reviewing design process models found in the literature, and it was found that some writers on design theory (Pahl & Beitz for example) prescribe an approach where, ideally, all possible design alternatives should be generated and numerically evaluated. However, it was concluded that this is not the way that designers work - in practice designers tend to re-use existing solutions, selected from their experience or from domain knowledge, and the initial solution is then repeatedly

"patched" and "repaired", unless it fails badly. The danger of obtaining a highly sub-optimum design by following this approach can be offset by the use of the "set-based concurrent engineering" model proposed by Ward and Seering, where sets of alternative solutions are developed concurrently, and the decision between them is taken as late as possible. In many of the design process models which were reviewed, the design process was characterised (formally or informally) as including a process of *refinement* - whether by the addition of information, the elimination of alternatives, the replacement of approximate values with more precise information or the replacement of abstract entities by more specific ones. Therefore an approach was required for the risk model which supports re-use of existing solutions and representation of historical data, which allows concurrent development of alternative solutions and which supports the representation of abstract entities.

The review of design process models, above, along with a review of current work on uncertainty modelling in design, yielded some insight into the types of uncertain information which are important during the early stages of the design process. A dual classification of uncertain information in early design was proposed, according to the nature (or type) and the subject of the uncertainty. Using this classification the types of uncertain information which a risk model should be able to represent were identified as including numerically and categorically uncertain information, incomplete information and abstract information - all of which may concern either the explicit, measurable, attributes of the artefact under design, or its environment. The risk model should also be able to represent numerical uncertainty in the (often approximate) methods which are used to evaluate "implicit design attributes" (such as performance parameters), and also categorical uncertainty due to the availability of a multiplicity of such methods for determining a given implicit attribute.

It was important that the risk model should support a design project risk management process. The eight key features of a project risk management methodology, which should therefore be supported by the risk model, were identified as: goal definition, risk identification, risk impact and probability evaluation, risk prioritisation, modelling risk relationships, mitigation and contingency planning, allocating and continually monitoring budget levels and continual risk monitoring.

12.1.3 Approach Taken in the Risk Model

A class-based, object-oriented (OO) approach was found to be particularly suitable for the risk model. Such an approach allows the representation of previous solutions and historical data using object classes and supports their subsequent re-use (either by inheritance or by component re-use) - this accords well with design practice and facilitates rapid model building. The ability to add objects to the model in a hierarchical fashion, as detail is added to the design, reflecting the constructive and incremental nature of the design process, was another benefit of the OO approach. There was also a need to represent abstraction and to support a refinement process which includes moving from the abstract to the concrete, for which purposes a class inheritance hierarchy may be employed.

A probabilistic representation for uncertainty was found to be the most suitable for the risk model. Fuzzy and interval approaches were considered, but the main outputs required from a risk model are essentially

measures of likelihood, rather than possibility. If the personalist view of probability as a measure of belief is adopted, then the designers are able to provide information suitable for use as input to a probabilistic model. Case studies of current early design practice based on particular design projects were conducted with Rover and Cegelec, to study the types of design domains where the risk methodology and tool should be applicable and to provide examples to test the generality of the modelling methodology. One finding in particular was that, at Rover, work begins on the configuration design - the different configurations of the vehicle which will be offered to the customer and how they are built from variant parts - during the early design stage. A model for the configuration design was developed for inclusion in the risk modelling methodology. The model allows the component manufacturing volumes, which have a strong impact on manufacturing costs, to be evaluated from estimates of the vehicle sales volumes for each product and customer option.

12.1.4 Modelling Methodology

A risk modelling methodology has been developed, based on OO principles, with a single-inheritance class hierarchy. A model is built by creating objects (termed Sim objects) which represent parts of the design, or manufacturing processes or materials, and editing their attribute values and the links between them. Objects are created by specifying a class - the class definition encapsulates knowledge about the represented category of object in the form of the names, types and default values of links and attributes, and also methods which are executable pieces of code which return a value for an attribute - possibly by using historical data. The methodology has been extended to represent uncertainty by allowing representation of probabilistic uncertainty in the object *attribute* values, in the values returned from firing *methods* and in the *links* (i.e. pointers) relating objects. Incompleteness is handled by using an explicitly defined value to indicate that attributes or links have unknown values. The methodology also allows the risk model to record multiple methods for determining an implicit attribute, ordered according to their accuracy as perceived by the modeller. The most accurate method for which the all the necessary input information is available is used to evaluate each implicit attribute. Thus new information can be used automatically as it becomes available.

The methodology provides three types of relationship, represented by object classes, which structure the risk model; an instance of ICO (or "is-a-component-of") relates a part to its component parts (for example an assembly to its sub-assemblies, or a task to its sub-tasks); an instance of IVO relates a part to its variants (used to model the configuration design); and an instance of IAO (or is-an-alternative-of) relates the source of a link (i.e. an object pointer), to each possible destination object. The IAO relationship class is the mechanism for representing uncertain links between objects. All three types of relationship may be specialised by deriving new classes which inherit from ICO, IVO or IAO.

12.1.5 Implementing the Risk Tool

Monte Carlo simulation was found to be the only suitable technique for propagating uncertainty in the attribute values through the methods, since it was required that both the input probability distributions of the attribute values and the methods themselves should be arbitrary, and also that there should be no accumulation of error as a result of passing the (uncertain) output generated by one method as input to another. Latin Hypercube sampling and integer sampling without replacement were chosen as the sampling strategies.

A software tool, RiTo, was built to support the risk modelling methodology. RiTo loads a risk model (stored in an ASCII text file) and allows the user to browse the model and to evaluate any attribute of any object by Monte Carlo simulation; RiTo then presents the simulation results as statistics (textually) and also graphically. Support for configuration design modelling is included by dividing the risk model into a *feature model* (containing the definitions of products and customer options which will be offered, in terms of features) and a *component model*, which contains all the components and assemblies which will be used to provide the features, with instances of IVO used to relate sets of variant parts (only one of which parts will be used in a particular configuration). RiTo can "prune" the Bill-of-Material hierarchy, by automatically making a selection from each set of variants, to represent the product and customer options of interest and can also automatically calculate the total production volumes for all the parts in the component model.

One of the statistics which RiTo can report is the probability that an attribute will achieve a specified "goal value", and conversely RiTo can also report the value which has a specified "goal probability" of being achieved. Thus, in a bidding situation, a project manager can allocate a goal probability to the cost of supplying the bid, and hence determine a supply value for use in determining the bid price which has an explicitly defined probability of being achieved.

12.1.6 Hierarchical Risk Sensitivity Analysis

Another key aim of the work was to support the project manager in choosing where best to concentrate resources in order to minimise the overall level of uncertainty. An uncertainty measure termed the *risk sensitivity* (RS) has been defined, which represents the sensitivity of the uncertainty in an output, $\boldsymbol{Y}$, to an input random variable. The risk sensitivity of $\boldsymbol{Y}$ to $\boldsymbol{x}$ is defined as the expected value of the reduction in variance of $\boldsymbol{Y}$ if $\boldsymbol{x}$ were to be replaced by a point value. This definition has the virtue that it can be applied to both categorical random variables (such as those which occur in IAO relationships) and numerical random variables. A numerical algorithm has been developed which uses a variety of techniques to evaluate an approximation to the risk sensitivities in a risk model, re-using the sample values generated during a Monte Carlo simulation wherever possible. It was proposed that the rank correlation coefficients should be used to provide an alternative uncertainty measure, but this is only applicable to the numerical random variables in a risk model. The purpose of both measures is to provide a ranking for the sources of uncertainty in the model.

The existence of a hierarchical containment structure in the risk model (the Bill-of-Materials structure) allows the risk sensitivities to be evaluated at a specified level of the hierarchy - by limiting the sources of uncertainty for which the risk sensitivity is calculated to those which are visible at a given level in the hierarchy, the problem of complexity in large risk models is addressed. The process of analysing the risk sensitivities at a high level in the containment hierarchy, and then repeating the analysis on the major source/s of uncertainty identified, was termed *hierarchical risk sensitivity analysis* (HRSA).

12.1.7 Evaluation of Tool and Methodology using Case Studies

The methodology and tool, and also the HRSA algorithm, were evaluated using the case studies of early design provided by the industrial partners. A risk model of a steel roughing mill, which included a

dependency from one branch of the containment hierarchy to another, was built and evaluated using RiTo. The results were compared with those obtained using a commercial Monte Carlo simulation package and found to agree well. A risk model of the interior trim area of a new vehicle development was built, and was used to provide examples to test the HRSA algorithm. For some of these examples, the results obtained for the HRSA (by using a spreadsheet to post-process statistics provided by RiTo) were compared with analytically calculated values, or values obtained using other more direct numerical methods. The HRSA algorithm was found to be adequate as a means of identifying the most critical uncertainties in a risk model, but to provide only approximate results.

The case studies showed that the risk modelling methodology had sufficient generality to represent designs in two diverse design domains and that the types of relationships provided for defining the risk model structure (IAO/ICO/IVO) met the modelling needs of the projects studied. The complexity of the interior trim model illustrated the need for simple summary metrics to guide decision-making and showed how they can be produced using such a model which provides metrics such as the standard deviation of total cost (or other attribute of interest), the probability of achieving the budget and the risk sensitivities.

12.2 Limitations

12.2.1 Limitations of the case studies

Case studies were only conducted at two companies - and only that at Rover was based on a live project. For reasons of commercial confidentiality, the author did not have access to the members of the design team during the Rover case study and thus was not able to report on the user responses to results obtained from risk model evaluations. The risk models were built by an intermediary (Neil Davis of the Warwick Manufacturing Group) who obtained the necessary information from the design team members. Also, no opportunities were presented to study the usability of the modelling technique itself - to investigate whether or not designers found it easy to construct risk models using the proposed methodology. It is at present quite time-consuming to build risk models, as no user-interface has been built for this purpose and the models must be built "by-hand", using a text-editor - therefore, even had access to the design team been possible, such studies would not have been possible. The Cegelec case studies were based on typical reported examples and were built by the author, so offered no opportunity for studying user responses.

The classes developed for the case studies made no use of "real" historical data, and were thus providers more of structural information than of numerical information. The encapsulation of historical data, and of heuristic and parametric methods which utilise this data, into object classes was originally perceived as one of the strengths of the methodology, but it did not prove possible to obtain the necessary historical data. It is important to acknowledge however that the over-interpretation of sparse or irrelevant historical data is just as serious a problem as a failure to use that historical data which is available. In the case studies, subjective probability judgements were used in place of historical data and this is a valid approach.

In principle, the risk modelling methodology which has been presented supports all of the eight key features of a project risk management process mentioned above - however, the intended mechanism for supporting

designers in risk identification has not been developed or exemplified. The incorporation of lists of discrete, uncertain, cost variations (accompanied by descriptive text) with each part (assembly, component, task, software module, etc.) would provide numerical support for the risk register advocated by writers on project risk management, and for the point-valued "risks" and "opportunities" currently recorded by designers at Rover. Default values of these lists, consisting of descriptive text with all the cost variations assigned a value of "Unknown" could be defined for each class, and would be automatically used when the class is instantiated. The designer would be obliged to explicitly enter a value for each element of the list (or to delete it) in order to obtain a value for the total cost of the part - thus the default values would serve as a class-specific checklist, aiding risk identification.

12.2.2 Limitations of the risk modelling methodology and tool

The most significant limitation of the risk modelling methodology presented in this thesis is its confinement to design domains where there is substantial re-use of existing solutions. In order to obtain sufficient benefit from using the risk modelling methodology to justify its cost in terms of extra time expended, the design company must be able to build up a repository of classes encapsulating knowledge about the types of entities which they design - if most of their work consists of "one-off" projects, with little foundation in previous designs, then the methodology will be of little or no benefit. Numerical data about existing solutions must be available in order to develop the classes for the repository - for example, historical data about the cost and weight of previously developed components. Analysis of this data and definition of classes incorporating suitable methods to utilise it may initially be time-consuming, but it is anticipated that this will be justified by the benefits realised when information can be obtained from historical data quickly and easily, during the design process.

A second limitation of the methodology is the absence of an explicit representation of time in the risk model - thus, for example, the model does not allow Rover to evaluate the possible future environmental effects of their design decisions. There is also no explicit representation of time (or design stage) during the design process itself. The modelling tool allows a sequence of risk models, representing the same artefact at different design stages, to be displayed and evaluated and compared. However, no time-stamps or model version numbers are recorded with the risk models and it is not possible, for example, to obtain a plot of the expected value of the total cost for the interior trim area against time, or to explicitly represent the design process. The process is supported, in the sense that the risk model can be modified to accommodate the changes which occur during the design process, but is not itself represented in the risk model.

There is a third limitation of the modelling methodology concerning the modelling of alternatives which must be noted. In the present implementation, it is not permitted under any circumstances for two alternative Sim objects to have the same bound sub-object - recall that bound sub-objects are both created and destroyed with their parent and hence a Sim object may only be bound into one parent. This provides a modelling difficulty in the case, for example, where two alternative assemblies share some components. At present the common components must be duplicated in each alternative. A solution to this problem has been conceived but not yet implemented.

Finally, the requirement that the `UncertainValues` in the risk model must be independent must be noted as an important limitation of the modelling methodology. If a statistical dependency exists between two attribute values and the modeller is unable or unwilling to extend the model to represent the cause of the statistical dependency as a causal dependency on a common lower-level attribute value, then there is no limit to the potential error which is introduced to the risk model. It would be possible to modify the simulation algorithm to allow the user to specify statistical dependencies between `UncertainValues`, however this would have an impact on the hierarchical risk sensitivity analysis algorithms, which would require modification.

Moving from the general features of the modelling methodology to more specific limitations, there are two limitations in the evaluation methods described which must be mentioned. The first concerns the sampling strategy used by RiTo, and the second concerns the hierarchical risk sensitivity analysis (HRSA) algorithm. The sampling strategy used when evaluating the risk model does not allow for the effect of IAO relationships (which are effectively branch-points in a decision tree) on the choice of samples, taken from continuous random variables, which are actually used to build up the output distribution. For example if 1000 iterations are performed and, because of its position in the decision tree, a particular continuous random variable has only a 20% chance of selection, then only 200 of its samples will actually be used in the evaluation. However, the random variable is initialised for Latin hypercube sampling by dividing its cumulative probability density into 1000 strata, of which a randomly selected 200 will be used to actually generate samples. Thus the benefits of Latin hypercube sampling and integer sampling without replacement, in enabling rapid reconstruction of the original PDF, are reduced for random variables which have a low probability of selection.

The second limitation in the evaluation methods described concerns the algorithm used for HRSA in the case when the inputs to the analysis are not independent and are known to have a non-linear relationship with the output. Suppose we wish to evaluate the risk sensitivity of $\boldsymbol{Y}$ to $\boldsymbol{x1}$ where $\boldsymbol{x1}$ is known to be correlated with $\boldsymbol{x2}$ and to be non-linearly related to $\boldsymbol{Y}$. It is not possible to perform a sub-simulation with $\boldsymbol{x1}$ fixed and $\boldsymbol{x2}$ varying, since the two are correlated - so the RS to $\boldsymbol{x1}$ cannot be determined by direct calculation. In this situation, the partial correlation coefficient $\rho_{Yx1.x2}$ is used to determine the RS value. The partial correlation coefficient represents the proportion of the variance in $\boldsymbol{Y}$ which is attributable to the linear regression of $\boldsymbol{Y}$ on $\boldsymbol{x1}$ with $\boldsymbol{x2}$ held constant. It is generally defined as the correlation between e_Y and e_{x1} where e_Y is the residual after subtracting the regression of $\boldsymbol{Y}$ against $\boldsymbol{x2}$ from the sampled $\boldsymbol{Y}$ values, and e_{x1} is the residual after subtracting the regression of $\boldsymbol{x1}$ against $\boldsymbol{x2}$ from the sampled $\boldsymbol{x1}$ values. When the relationships are not linear, this will clearly yield inaccurate results, as was illustrated in Case 5 of Chapter 11. It would be possible to use a second order or higher order polynomial for the regressions, and this would improve the accuracy of the results at the cost of considerably greater computation time. It could be argued that this level of accuracy is unlikely to be required in an analysis whose overall aim is simply to identify the critical uncertainties, rather than providing an accurate risk sensitivity measure for each input attribute.

12.3 Future Areas of Research

12.3.1 Validation of the Novel Features of Tool and Methodology

Although the case studies have been used to validate the risk model evaluation methods and to test the generality of the modelling structure, the more general novel features of the methodology cannot yet be said to have had their usefulness proven. It will not be easy to validate the concepts of object-oriented risk modelling, of continual refinement of a design risk model or of the integration of design risk management and project risk management into the design process. Obtaining suitable case studies will be difficult - this will require full access to the design team during a "live" design project and they will incur a large time overhead, partly in order to build the risk model but mainly because of the need to be observed and to provide feedback. Interpretation of the results will also be difficult - it is rarely possible to present empirical proof that the benefits of risk management outweigh its costs. It is not possible to distinguish good luck from good management based on a single project, yet there is unlikely to be an opportunity to compare a reasonably large sample of projects conducted with and without the risk modelling approach.

12.3.2 Risk Modelling in a Team-Based Environment

There are many important issues concerning the use of the risk modelling methodology in a team based environment which have not been addressed in this work. The object-based and hierarchical nature of the model lends itself to a multi-user implementation - for example presenting the possibility for encapsulation of private data within objects, and providing the model legibility and transparency to other users which is absent, for example, in spreadsheet-based probabilistic models. Nonetheless, there are difficult issues of security, remote access and concurrent access which must be addressed if the methodology is to become a tool for co-operative working. If very large risk models are built, which is likely if all the team-members are able to co-operate over the building of a single risk model, and if the designer must obtain the simulation results over a network or the Internet, then simulation and data transfer times may become unacceptably slow. In this case, the possibilities for performing partial simulations, storing simulation results for each portion of the risk model separately, and re-using simulation results wherever possible, only re-simulating the portion of the risk model which has changed, may need to be explored.

In addressing the security issues in a multi-user environment, the various roles involved in developing the risk model must be considered, along with their associated responsibilities. A designer may have responsibility for the structure of the assembly on which he is primarily engaged, for certain of its attributes and for its sub-objects. The team leader may be responsible for assigning the budget levels within her area. A cost estimator may be responsible not only for maintaining the historical data and heuristic methods for the classes in which he is an expert, but also for personally providing cost estimate judgements which are stored into assemblies and components which are "owned" by other designers.

12.3.3 Integration with Existing Databases

It is likely that much of the information required for the risk model will already be stored in other company databases, such as product modelling packages, electronically stored costing forms etc. The difficulties of maintaining parallel sets of data and propagating changes between them mean that integration of the risk

model with information stored in existing databases may prove essential. Methods for extracting the structural data required for the risk model from existing databases or alternatively for incorporating the ideas presented here into existing product modelling tools should be investigated.

12.3.4 Distinguishing Alternatives Which are Under the Control of the Designer

The same representation is currently used in the risk model for choices between design alternatives which are under the control of the designer (such as a choice between two different manufacturing processes) and for uncertainty regarding which of a set of alternative objects will come into play, when the choice is beyond the designer's control and thus forms part of the design environment - for example, uncertainty regarding whether the preferred option of two alternative materials will be permitted by future legislation. The implications of distinguishing these two types of IAO on decision making should be investigated, as this would allow the designer to explore the results of conservative decision-making for those alternatives which are under his control on the overall risk levels. For example, faced with a choice between an established manufacturing process and a newly developed process whose use may have abstract benefits in the future - such as gaining expertise for the company in a new technology area - the designer may like to attempt to quantify the additional risk she is accepting in order to accrue those abstract benefits.

12.3.5 Development of Numerical Classes

Further research is needed to verify the feasibility of transforming "raw" historical data into the form required for representation as risk model object classes. "Numerical" classes should be developed, containing numerical default values (as probability distributions), historical data and parametric and heuristic methods which utilise this data. A promising area for development of such numerical classes is whole life costing for early design - prediction of the whole life costs and environmental impact of a designed artefact is a problem which by its nature concerns predicting the future and hence contains much uncertainty and also one where much numerical data is available. Life cycle assessment (LCA) is a technique whose use is currently growing and which is usually applied at the detail design stage. In LCA studies, an inventory of the product is built which includes every environmentally relevant input or output during the product lifecycle. This yields an environmental profile of material and energy consumption and emissions, and the impact of this profile on the environment is then evaluated numerically. Tools also exist for whole life costing in early design, but they tend to use highly simplified models, and are, like LCA, deterministic. Probabilistic uncertainty could be incorporated into whole life models by using existing numerical data on materials and manufacturing processes (some of which is utilised in LCA studies), to develop numerical classes. This would allow the designer to obtain an indication of the likely environmental impact of the design during the early design stage. In developing classes for whole life costing, it may prove beneficial to introduce explicit modelling of time into the risk model.

12.3.6 Combining Evidence from Multiple Attribute Derivation Routes

Finally, during the work reported in this thesis, investigation into means of combining evidence from multiple derivation routes was abandoned, as it was felt by the industrial partners that combining evidence in this way would give rise to results which were difficult to audit or justify and were not traceable. Nevertheless, the author feels that this would be a fruitful area for future work - the combined results could

be presented in addition to the results obtained from the "live route", as at present. Provision of a measure of the degree of variation between the results obtained from different routes and flagging of inconsistencies would also be beneficial.

12.4 Significance of Research on Original Problem

The overall goal of the work presented in this thesis was to provide new tools to support design engineers and their managers in decision-making during the early stages of the design process. This goal has been approached by the development of a risk modelling methodology and prototype software tool (RiTo). Although RiTo is not a polished and completed software package, lacking in particular a modelling interface and multi-user capabilities, it embodies the ideas presented in the thesis. RiTo's use in case studies conducted with industrial partners successfully demonstrated both the generality of the methodology and also the strength of the risk model evaluation methods which have been described.

The aims of the research, presented in Chapter 1, have largely been met; evaluation of the risk model using goal values and goal probabilities enables the project manager to explicitly take risk into account when setting bid prices; a designer may compare not only the expected values of cost, weight etc., but also the level of risk present for each alternative when facing a choice between design alternatives; a "hierarchical risk sensitivity analysis" algorithm has been demonstrated (though not fully implemented in RiTo) which supports the project manager in determining the critical uncertainties in the risk model and hence where best to concentrate resources; and the project manager is able to monitor overall risk levels using graphics and statistics provided by RiTo.

Successful risk management does not involve removing all risks - it involves knowing the level of risk you are accepting when you take a decision. It is hoped that this work will contribute towards helping designers and their managers to make better decisions, informed by an understanding of the level of risk involved.

12.5 Publications Arising from the Work Reported in This Thesis

12.5.1 Refereed Papers

[Crossland 1995] Crossland, R., Sims Williams, J. H., and McMahon, C. A., "An Object-Oriented Design Model Incorporating Uncertainty for Early Risk Assessment", *Proceedings of Computers in Engineering*, ed A. A. Busnaina, ASME, Boston, pp.93-112, September 1995.

[Crossland 1995] Crossland, R., Sims Williams, J. H. and McMahon , C. A., "An Object-oriented Design Model Incorporating Uncertainty for Early Risk Assessment", (Poster), *Proceedings of the International Conference on Engineering Design*, (ICED'95), Praha, Czech Republic, pp.1555-1556, August 1995.

[Crossland 1997] Crossland, R., Sims Williams, J. H., and McMahon, C. A., "Risk Assessment Using RiTo: An Object-Oriented Risk Modelling Tool for Early Design", submitted to *Research in Engineering Design*, February 1997.

12.5.2 Public Presentations

[Crossland 1996] Crossland, R. and Sims Williams, J. H., "An Object-Oriented Risk Model for Early Design", *Evening Lecture to the Bristol Branch of the British Computer Society*, final lecture in a series of eight entitled "Software Estimation and Risk", presented by R. Crossland, March 1996.

12.5.3 Project Reports

Report number STARTED/BRI/003/1, Crossland, R., "Example Case Study of Early Design Information Modelled Using the Fusion Method", April 1994.

Report Number STARTED/BRI/004/1, Crossland, R., "Spreadsheet Cost-Risk Demonstration Software", May 1994.

Report Number STARTED/BRI/005/1, Crossland, R., "Review of Risk and Uncertainty in Engineering Design Models", July 1994.

Report Number STARTED/BRI/006/1, Crossland, R., "Requirements for GUI Toolkit to Build Rrisk Assessment Model", July 1994.

Report Number STARTED BRI/007/1, Crossland, R., "Current Practice in Risk Assessment", October 1994.

Report Number STARTED/BRI/008/2, Crossland, R., "Requirements Specification for Risk Assessment Tool", November 1994.

Report Number STARTED/BRI/009/1, Crossland, R., "Using the Fusion OO Methodology with C++ and Windows", February 1995.

Report Number STARTED/BRI/010/1, Crossland, R., "Theory for Multi-Dimensional Output and Figure of Merit", April 1995.

Report Number STARTED/BRI/011/2, Crossland, R., "Schema for Rover Case Study", January 1996.

Report Number STARTED/BRI/013/1, Crossland, R., "Evaluation of Version 1.0 Risk Tool", January 1996.

Report Number STARTED/BRI/014/4, Crossland, R., "User Guide for Version 2.0 of Risk Tool", April 1996.

Report Number STARTED/BRI/015/1, Crossland, R., "Evaluation of Version 2 Risk-Cost Software", July 1996.

Report Number STARTED/BRI/016/1, Crossland, R., "Version 2.1 of Risk Tool", July 1996.

Report Number STARTED/BRI/017/1, Crossland, R., "Configuration Modelling Under Uncertainty", July 1996.

Report Number STARTED/BRI/018/1, Crossland, R., "A Simple Risk Model of a Dynamic Positioning System", July 1996.

Report Number STARTED/BRI/020/1, Crossland, R., "Version 2.2 of Risk Tool", October 1996.

Report Number STARTED/BRI 021/2, Crossland, R., "Risk Sensitivity Analysis in an OO Hierarchical Risk Model", December 1996.

Glossary

abstract class

In C++, a class which contains one or more *pure virtual methods* and therefore cannot be instantiated directly. Only *sub-classes* which contain an implementation for all the pure virtual methods can be instantiated.

attribute

In RiTo, an attribute of a *Sim object* is base-typed - it may have type FLOAT, Str, INTEGER or BOOL.

auxiliary model

In the view taken of design information models, an auxiliary model contains information which is required in addition to the *explicit design attributes* in order to deduce the *implicit design attributes.* An example would be a Finite Element Analysis model.

base classes

In RiTo, the base classes are the "building block" classes supplied with the risk toolkit.

bill-of-materials

A list of all the parts required to manufacture an artefact. In RiTo, the bill-of-materials view of the *risk model* is a simplified view which shows the assemblies and components arranged in a hierarchy but excludes, for example, materials and manufacturing processes.

BoM

Bill-of-materials.

call-back function

In C and C++, a function whose address is passed (as a function *pointer*) to another piece of code which can then invoke (or "call-back") the function as and when it is required.

CDF

Cumulative probability distribution function.

constructor

In C++, a special method, part of the class definition, which is only called when an instance of the class is created.

container object

A *Sim object* which is a node in a bill-of-materials type decomposition of the *risk model.* Typically, assemblies and components.

CustomerOption

In Rover terminology, an extra feature or set of *features* which the customer may choose to purchase. Each *product* has a set of valid customer options which are available to the customer. In RiTo, CustomerOption is one of the base classes.

data-member

In C++, the data associated with a particular class. A data member may a value or a pointer, or an array thereof, or a reference. It may be a base-typed value (or a pointer or a reference to a base-typed value) or it may be an object value (or a pointer or a reference to another object).

DDE

Dynamic data exchange. A simple mechanism provided by Windows 3.1 for data sharing between applications. Used by RiTo to provide data to an Excel spreadsheet for graphical display.

derivation route

In RiTo, a means of obtaining a value for an attribute of a *Sim object*. This may be simply a point value, but it may also be a pointer to an *UncertainValue* object or a method or attribute of another Sim object reachable by traversing links.

destructor

In C++, a special method, part of the class definition, which is only called when an instance of the class is created.

deterministic method

In RiTo, a deterministic method is a method which may take either uncertain-valued or point-valued attributes as inputs and which calculates a point value as output. A deterministic method is fired many times during a simulation, if any of its input attributes are uncertain.

direct inheritance

A sub-class inherits directly from a super-class (i.e. is an *immediate sub-class* of the super-class) if there are no intermediate classes in the inheritance graph.

DLL

Dynamically Linked Library - see *dynamic linking*.

dynamic linking

When using a compiled language such as C or C++, it is generally necessary to link together several modules of code to build an executable program. If this linking is performed dynamically, at run-time, it is termed dynamic linking (as opposed to *static linking*). Under the Windows operating system, it is necessary to build a special type of library (code module) for dynamic linking, termed a *DLL* or *dynamically linked library*. Any changes made within a DLL are thus entirely isolated from the remainder of the code - a DLL may simply be replaced with a modified version which will then

be called by the main program when it is invoked. The main program does not require any recompilation or explicit re-linking.

Editable object

In RiTo, an Editable object may be stored in the *object repository file* and displayed using the risk tool. Editable is the root class of the *base classes*.

explicit design attribute

The designer explicitly specifies the design in terms of explicit attributes. An objective, measurable, physical property of the artefact under design taken in isolation. An example would be the physical dimensions of a mechanical part.

Feature

In Rover terminology, a feature describes the capability of the product in terms of function - a feature generally has both a price and a cost associated with it. In RiTo, Feature is one of the *base classes*.

generic PhysicalObject

In RiTo, a part which has more than one *variant* included in the *risk model*.

Hierarchical risk sensitivity analysis

A technique where the *risk sensitivity* of the output of interest to each input attribute is evaluated at a given level of a "containment" hierarchy in the risk model (based on the *bill-of-materials* hierarchy). The input attribute which is found to make the major contribution to overall uncertainty then has its own risk sensitivity to each of its inputs explored at a lower level of the containment hierarchy, and so on.

heuristic method

In RiTo, a heuristic method takes point values as inputs and produces an uncertain value as output - thus the method itself is a source of uncertainty in the model.

HRSA

Hierarchical risk sensitivity analysis.

IAO object

In RiTo, an "is-an-alternative-of" relationship, relating a set of *Editable objects* only one of which will eventually be chosen for inclusion in the designed artefact. Such relationships are represented by objects of class IAO or of a class derived from IAO. IAO is one of the *base classes*.

IAO parameter class

In RiTo, the minimum class of *Sim object* related by an *IAO* object. An IAO object may relate objects whose class is the same as, or derived from, the IAO's parameter class.

ICO object

In RiTo, an "is-a-component-of" relationship, relating for example an assembly and its components or a task and its sub-tasks. Such relationships are represented by objects whose class is derived from ICO. ICO is one of the *base classes*.

IVO object

In RiTo, an "is-a-variant-of" relationship, relating a set of *variant* objects. Such relationships are represented by objects whose class is derived from IVO. IVO is one of the *base classes*.

immediate sub-class/super-class

Class A is an immediate sub-class of class B if A inherits *directly* from B. Similarly for super-classes.

implicit design attribute

An implicit design attribute is an evaluation of the design according to financial, technical or other criteria. The values of implicit design attributes generally depend upon the interaction of the explicit attributes with the design environment (e.g. usage or loads), and must be predicted using a model of some sort or by testing a prototype.

interaction effect

An interaction effect exists between two inputs *xi* and *xj* when the effect of *xi* upon *Y* depends upon the value of *xj*. This is an example of a two-way interaction effect. In a three way interaction effect, the effect of *xi* depends on the values of both *xj* and *xk*, and so on. An interaction effect arises, for example, from the cross-term a_{12} in a model of the form:

$$Y = a_1 x1 + a_2 x2 + a_{12} x1 x2$$

interface attribute

In RiTo, an attribute which belongs (directly or indirectly) to one container object and is also used by one or more other container objects.

link

In RiTo, a link from a *Sim object* is an object-valued property - a pointer to another Sim object.

live route

In RiTo, the most preferred (highest priority) derivation route for a *Sim object attribute* value which can return a value without encountering any UNKNOWN attribute or *link* values.

method

In C++ (and other OO programming languages) a function which is associated with a particular class. A C++ method may, or may not, return a value. In RiTo, a method is an executable piece of code provided to a *Sim object* by its class, which calculates and returns a value for an *attribute.* RiTo supports *deterministic methods* and *heuristic methods.*

object id

In RiTo, a unique identifier for each object (*Sim* or *Editable*) stored in the *object repository file* (unique within that file). Used internally by RiTo.

object repository file

In RiTo, the file in which the *risk model* is stored. A human-readable, ASCII text formatted file containing information structured according to object identity. Usually stored as a file with extension .SIM.

OODB

Object-oriented database.

OOP

Object-oriented programming.

PDF

Probability density function (sometimes also used to mean probability distribution function, but the meaning should be clear from the context).

pointer

A language construct in C and C++; the address in memory of a variable, object or function or method, or of another pointer.

PhysicalObject

In RiTo, PhysicalObject is one of the *base classes.* It is used to represent any component or assembly in the *bill-of-materials.* In Rover terminology, an instance of PhysicalObject is a "part".

Product

In Rover terminology, a "vehicle offering"; a particular configuration of the vehicle, which a customer may purchase. There will be a limited number of products offered - these are the "brochure models" or standard offerings. In RiTo, Product is one of the *base classes.*

pure virtual method

In C++, a *virtual method* which does not have an implementation in the class in which it is defined.

refinement

A characterisation of the design process. Refining processes include the addition of information, decreasing levels of abstraction and movement from the approximate to the precise.

risk model

A design model incorporating uncertainty. In RiTo, a risk model consists of *Sim object* instances stored in the *object repository file*.

risk sensitivity(RS)

The risk sensitivity of Y to x is intended to provide a measure of the amount of the observed variance in Y which is "due to" variance of x; thus the risk sensitivity is a combination of the sensitivity of Y to x and the variance of x. More precisely, it is hoped to provide some measure of the reduction in output variance which could be expected if x were to be replaced by a point value. The risk sensitivity is thus defined as the expected value of the reduction in variance of Y when x is replaced by a point value.

RS

Risk sensitivity.

rv

Random variable (shown in italicised bold script in this thesis e.g. $\boldsymbol{Y}$).

sample value

In Monte Carlo simulation, a single value which may be taken by an *rv*. The value is sampled from the *PDF* of the *rv*.

sample

In Monte Carlo simulation, a set of *sample values* for an *rv*. The distribution of the sample is given by the *PDF* of the *rv*.

schema evolution

In an *OODB*, the ability to modify the schema (class definitions) when instances of the classes already exist; the existing instances are automatically modified to comply with the new class definitions.

schema file

In RiTo, a file called SCHEMA.TXT which resides in the same directory as the risk tool executable (RITO.EXE) and which contains the class definitions. It always contains the *base class* definitions and may also contain user-specific classes derived from the base classes.

sensitivity

The sensitivity of a model output Y to an input x is a measure of the size of the change in Y which would result from a unit change in x, about the current value of x. In a stochastic model, the sensitivity must be defined as the expected value of the change in Y which would result from a unit change in x.

Sim object

In RiTo, an object in a *risk model* representing part of a design. Sim is the root class of the RITO *base classes* - thus all RITO object classes are derived directly or indirectly from Sim (with the exception of *UncertainValue* and *IAO* classes). A Sim object may have uncertain attribute values, alternative link values and may have multiple derivation routes defined for any of its *attributes*.

static linking

When using a compiled language such as C or C++, it is generally necessary to link together several code modules to build an executable program. If this linking is performed once, at compile-time, to yield a single monolithic executable program, it is termed static linking (see *dynamic linking*).

sub-class

In *OOP* terms, B is a sub-class of A if B inherits data and methods from A; i.e. if B is below A in the inheritance hierarchy (for single inheritance) or inheritance graph (for multiple inheritance). In OO modelling terms, B is a sub-class of A if B is a specialisation of A; if the set of objects in class B are a sub-set of those in class A.

super-class

In *OOP* terms, A is a super-class of B if B inherits data and methods from A; i.e. if B is above A in the inheritance hierarchy (for single inheritance) or inheritance graph (for multiple inheritance). In a single inheritance hierarchy each class has a maximum of one super-class. In a multiple inheritance hierarchy, a class may have several super-classes. In OO modelling terms, A is a super-class of B if B is a specialisation of A; if the set of objects in class B are a sub-set of those in class A.

top object

In RiTo, an *Editable object* with *object id* 0, stored at the beginning of the *object repository file*. It is the root of persistence.

UncertainValue object

In RiTo, an object, stored in the *object repository file*, which represents a probability distribution function. The attributes of such an object are the parameters of the probability distribution and its class is derived from UncertainValue. For example, an instance of TriangleFloat is an UncertainValue object with attributes min, likely and max.

variant

In RiTo, a *Sim object* (usually a *PhysicalObject)* which belongs to a related set of variants. The definition of such a set is that they will all be manufactured but a maximum of one member of the set will be included in any one particular product instance. Also, all the members of a set of variants fit into the *BoM* hierarchy in the same place.

virtual method

In C++, a *method* which may be overridden (re-implemented) in a *sub-class*.

Appendices relating to

RISK IN THE DEVELOPMENT OF DESIGN

LIST OF APPENDICES

Appendix A: Requirements for Risk Assessment Model and Tool

Introduction to the Appendix

This appendix contains a description of the requirements which were identified for the risk model and for the toolkit with which the risk model is browsed, edited and evaluated. A requirements specification document was produced towards the beginning of the project (Report number STARTED/BRI/008/2), and this is reproduced almost verbatim in this section, other than some minor changes to the grammar and the page layout. In addition to these minor changes, each paragraph in the requirements specification document has also been augmented with a code indicating the final status of the requirement at the end of the project and in some cases further notes have also been added for clarification. The status code and optional notes which have been added are shown enclosed within a black border to distinguish them from the original document, and the notes are shown in italics. The status codes are:

- **Met** - the requirement described in the paragraph is met by the risk toolkit as it is currently implemented.
- **Abandoned** - during the course of the project, it was found that the requirement no longer existed.
- **Transformed** - during the course of the project the requirement was transformed, and the transformed requirement is met by the risk toolkit.
- **Not implemented** - the requirement is still relevant, but has not been implemented due to constraints of time and/or resource.

Reasons for abandoning or transforming requirements include discovering that a requirement was inconsistent with other requirements and discussions with the industrial partners revealing that the requirement did not meet a true need. There are many ideas contained in this section which have not been implemented in the risk toolkit, but some of these ideas are in themselves a valuable output from the research and are included here for that reason.

Requirements Specification Document for Risk Assessment Tool

Abstract

This document describes the full set of requirements which have been expressed by the industrial and academic partners in the STARTED project for a risk assessment toolkit. It is not expected that the software which is delivered at the end of the project will fully meet all of these requirements - it is expected that a sub-set of these requirements will be satisfied by the toolkit which is delivered.

The aim of the risk toolkit is to allow the users to build a model which captures all of their uncertain knowledge about a design, during the early stages of the design process. The risk toolkit may then be used to evaluate this model and produce an assessment of the risks.

This document describes the certain and uncertain data-types which the model is required to represent and the relationships between the data items (including constraints, derivation of data values and representation of alternatives). The requirements for the method of evaluation of the model and for the incorporation of historical data are described. The types of graphical and tabular output which are required are defined. This document also describes the requirements for tools to build, browse and edit the model.

Contents

A.1 Introduction

A.1.1 Purpose

A.1.1 This document defines the requirements for a risk assessment tool for the STARTED project. Each paragraph in this document has been assigned a unique number - the paragraph numbers are intended to provide a point of reference for discussion of the requirements amongst the STARTED partners. Where a requirement may be attributed to a particular industrial partner, a three letter code enclosed in square brackets indicating which partner has expressed the requirement follows the paragraph number. The three letter codes are:

[CEG] (CEGELEC Projects Ltd)
[ROV] (Rover Technology)
[OVE] (Ove Arup and Partners)

A.1.2 Scope

A.1.2 This document describes the full set of requirements which have been expressed by the industrial and academic partners in the STARTED project for a risk assessment toolkit. It is not expected that the software which is delivered at the end of the project will fully meet all of these requirements - it is expected that a sub-set of these requirements will be satisfied by the toolkit which is delivered.

A.1.3 Overview

A.1.3.1 An object-oriented view is taken of the information in an early design model. The model consists of a database of instances of classes (i.e. design objects) with relationships between them. The evolution of the model during the process of design involves the creation of new objects as increasing detail is added to the model and the modification of object attributes and deletion of objects as design decisions are taken. It also involves the creation of new *classes* of object, specialisations of general classes of design object. The general classes will be re-used from one design to the next, whereas the specialisations may be specific to a particular design. A set of super-classes will be provided with the system, which each user-company will specialise to suit their own particular product area and design process. Examples of such super-classes are `PhysicalObject`, `Management` and `Design`.

Met / Abandoned : *There was found to be no requirement for super-classes for Management and Design The super-classes were termed the "base classes" in the implementation of the risk tool.*

A.1.3.2 It is anticipated that teams of co-operating designers will build and modify the design model. A modeller will be responsible for creating new specialised classes and modifying the existing class definitions. It is anticipated that, initially at least, a single modeller will be responsible for building all of the classes.

Met :

A.1.3.3 The risk assessment tool will enable uncertain information about the early design of a product to be modelled. From this information, and the relationships between the pieces of information, the tool will produce estimates of risks in the project. It will identify which uncertain parameters have the largest effect on the risk. The tool will be used during the early design phase of the project, and the uncertain information will be modified by the users (generally rendered more certain and more detailed) as the design progresses. The tool will store information on the progress of the design in an archive. A historical evidence tool will also be provided which will use the archive to produce estimates of uncertain design parameters.

Met / Not Implemented : *The historical evidence tool was not implemented.*

A.2 Overview Of Requirements

A.2.1 The designer will be able to sit in front of a screen and add new instances of physical objects, management, design and other super-classes as the design proceeds. He will be able to modify the uncertain and certain attributes of these objects.

Met / Not Implemented : *A mouse-driven model-editing interface was not implemented, the risk model is built as an ASCII text file.*

A.2.2 The designer will be able to generate new models by copying instances from existing models and combining them or by removing some instances from an existing model.

Met :

A.2.3 The modeller will be able to add new sub-classes and add new attributes and methods to existing classes by writing C++ code and re-compiling the risk assessment application, during the design process. The existing database will be re-configured into the new schema.

Met :

A.2.4 A code-generation tool will be provided which will assist the modeller by freeing him from the need to write repetitive C++ code and helping to ensure that the new class definitions are consistent.

Met :

A.2.5 The designer will be able to create and delete relationships between the objects in the database - constraints on attributes and objects, dependencies between attributes and objects, aggregation of objects into a BOM hierarchy.

Met / Transformed : *The requirement to support general modelling of constraints was transformed into the simpler requirement to model goal (or "target") values for attributes. This was driven by the realisation that general constraint modelling was taking the risk modelling tool away from its intended domain of decision support and into the domain of automated design.*

A.2.6 The system will provide a risk analysis at any stage as required.

Met :

A.2.7 All the information in the system should be accessible and perusable from a terminal by point and click. It is assumed that this will be done using indented lists to illustrate hierarchical relationships between objects, attributes and classes.

Met :

A.2.8 When browsing uncertain attributes, if the data has been changed since the last simulation (i.e. if the net is eligible for execution) then point values will be displayed which indicate the likely or expected value for the attribute after a simulation.

Met : *Indicative point values are always displayed, and the user may choose between an approximation to the most likely value and an approximation to the expected value.*

A.2.9 The system will contain and allow the use of historical cost and resource data.

Met : *Historical data, and methods to utilise it, can be included as part of the definition of a class.*

A.2.10 The system will combine information from different sources to produce its risk analysis. The system will automatically select and use different information sources and methods depending upon what information is available in the model. The system will generate a user-readable audit-trail, identifying the sources of the information used at each level in the simulation.

Met :

A.2.11 Warnings will be generated if:

- Differing information sources give widely different results
- Constraint violations occur

Not Implemented :

A.2.12 The system will be able to determine the area which makes the greatest contribution to the overall risk. This feature will help a risk assessor to determine where investment is most likely to result in reducing the uncertainty.

Met :

A.2.13 The system will contain alternative designs and will produce a comparison of the risk for each alternative.

Met :

A.2.14 Each piece of information stored in the system will have an owner, and only the owner may modify the information.

Not Implemented :

A.3 Uncertain and Certain Data Types

A.3.1 The types of uncertain attribute values supported will be

- interval values
- a selection of analytically defined probability density functions (PDFs)
- numerically defined PDFs

Transformed / Met : *Interval values are represented in the model by a Uniform PDF. Any piece-wise linear PDF may be represented as may Triangular, Beta, Normal and other analytically defined PDFs.*

A.3.2 Fuzzy attribute values will not be supported, as risk assessment concerns *likelihood* and the extension principle for fuzzy numbers does not correctly predict the likelihood of a particular value being attained.

Met :

A.3.3 For an interval value, the minimum and maximum values taken will be specified by the designer:

interval(<min>, <max>)

and where possible interval arithmetic will be used to propagate the uncertainty through methods and dependencies.

Abandoned : *The requirement for the model to contain both interval representations and probabilistic representations and to use different uncertainty propagation techniques for each (interval arithmetic and , for example, Monte Carlo simulation) was abandoned for two reasons. Firstly, it was concluded that the interpretation of results obtained from such a hybrid system would be difficult, because their meaning would be unclear. However, if the intervals are initially replaced by Uniform distributions and simulation is used throughout the model, then the results obtained simply represent likelihood in the standard probabilistic sense, subject to the assumption that all values within an interval are equally likely. Secondly, the design decision to adopt "black-box" methods (see 5.1 below) meant that the methods could not be guaranteed to be monotonic and thus an interval propagation technique would need to take samples from within each interval, not just from either end. Thus simulation would be the only computationally feasible option.*

A.3.4 Simulation will be used to propagate the uncertainty in PDF valued attributes through methods and dependencies. Examples of analytically defined PDFs which will be supported are:

triangle(<min>, <likely>, <max>)
normal(<standard deviation>, <mean>)
lognormal(<standard deviation>, <mean>)

Met :

A.3.5 Numerically defined PDFs will be defined, for example, by :

histogram(<min>, <max>, <number of bars>, <height of first bar>,...,<height of last bar>)
discrete(<number of values>, <first value>, <probability of first value>,...
,<last value>, <probability of last value>)

dice(<number of values>, <first value>, ..., <last value>)

The values given in the *dice* PDF above are assumed to all be equally probable (like the set of possible outcomes from throwing a pair of fair dice).

Transformed / Not Implemented: *The requirement to support numerically-defined PDFs was met by including support for any piece-wise linear PDF and for discrete-valued PDFs. Explicit provision of an equi-likely discrete PDF was not implemented.*

A.3.6 If it is necessary to combine interval-valued and PDF-valued attributes then the interval will be represented as a rectangular PDF and simulation will be used.

Abandoned : *The requirement to support interval-valued attributes was abandoned.*

A.3.7 The base-types of all uncertain and certain attributes may be *string*, *real*, *integer*, *enumeration types* (which may be user-defined) or *instance identifier*. An example of an enumeration type is BOOLEAN which takes values TRUE or FALSE. It will not be permitted to use enumeration types or strings in a context where ordering is expected (for example as the limits of an interval).

Met / Not Implemented : *Enumerated types were not implemented. Boolean values were implemented as a distinct base type.*

A.3.8 Attributes with a base-type of *instance identifier* will be used to represent pointers to other objects. This will enable relationships between objects to be themselves represented as objects. For example, a model may contain three objects :- `car` (an instance of class `Car`), `wheel` (an instance of class `Wheel`) and `components_of_car` (an instance of class `IsAComponentOf`).

Met : *Pointers to other objects were termed "links", and the term "attribute" was reserved for string/real/integer/Boolean valued attributes.*

A.3.9 Attributes will also be permitted to take values which are *lists* (ordered) and *sets* (un-ordered) of the base types mentioned above.

Not implemented : *General collection classes were not implemented. Links and attributes are assigned a cardinality in the class definition, which may be greater than one and may be variable, so "arrays" can be modelled. However, the "array" does not have its own object identity outside the parent object.*

A.3.10 [CEG] The system will also support *user-defined spreads*. A user-defined spread will define the remaining parameters of an interval or a PDF as a function of the estimated value. For example,

quite_sure(x) = triangle(0.9 * x, x, 1.1 * x)

where x is the estimated value.

Not implemented :

A.3.11 [ROV] The system will be supplied with pre-defined base classes with built-in methods and attributes. The user will be able to re-name these built-in methods and attributes (without needing to modify the class definitions themselves or re-compile the risk assessment tool) and assign them names which are meaningful to his particular application.

Not implemented : *Re-naming of classes, attributes, links and methods was not implemented.*

A.4 Constraints

The system will support three kinds of constraint:

A.4.1 [ROV] Structural selection constraints.

A.4.1.1 The creation deletion of a particular object constrains other objects (which could be anywhere in the model) to be created/deleted. For example, the creation of an object of class `AutomaticTranmission` constrains any object of class `ClutchPedal` to be deleted. The system will support this type of constraint by offering to create/delete objects automatically when the user creates/deletes an object. Such constraints will be defined as part of the class definition. If the design model includes several alternative designs, the system will only check for the existence of the constrained objects within the alternative which is being edited.

Structural selection constraints may also involve attribute values:

A.4.1.2 The creation/deletion of an object may change the permitted values for an attribute elsewhere in the model.

A.4.1.3 The value taken by an attribute may constrain other objects to be created or deleted.

A.4.2 [ROV] Attribute absolute constraints. An attribute may have a limited set of permitted values which do not depend upon anything else in the model. If the designer enters a value which violates, or is likely to violate, or may possibly violate (the entered value may be uncertain), such a constraint, he will be immediately warned. If the constrained attribute is not directly entered by the designer, but derived from elsewhere in the model, then the system will calculate the probability that the constraint will be violated and if necessary a warning will be generated after a simulation has been performed.

A.4.3 [ROV] Attribute interaction constraints. An attribute value constrains the value that other attributes (possibly of other objects) may take. If the designer enters a value for an attribute which is participating in such a constraint, then the system will immediately check the entered value against the indicative point values (see paragraph 9.4.1) for the other participants. A warning will be generated if the entered value is likely to violate the constraint. When the model is simulated, the system will calculate the probability that the constraint will be violated and if necessary a warning will be generated.

A.4.4 The system will support both types (attribute absolute and attribute interaction) of attribute constraint by providing a base class `Constraint`. The designer may create one or more instances of this class at run-time.

A.4.5 The system will not attempt to automatically modify attribute values so as to ensure that the constraints are satisfied.

A.4.6 The designer may specify a probability threshold, above which a warning will be generated. The system will provide a default probability threshold, which the designer may optionally edit.

A.4.7 The designer will also need to specify *to whom* the violation should be flagged - which of the constrained attribute values must be changed (by the owner of the attribute) in the event of a violation.

A.4.8 The attribute value derivation mechanism (see A.5 Attribute Value Derivation, below) provides another means for the designer to constrain values in the model. For example, if the designer wishes to constrain two attribute values to be equal, he may simply derive one value from the other.

Abandoned : *It was concluded that meeting these requirements would involve building an automated design tool rather than the decision support tool originally envisaged. It was also concluded that definition of such constraints would only be possible in highly static design areas - and that this would not generally be possible in early design.*

A.5 Attribute Value Derivation

A.5.1 The model consists of objects which are instances of the supplied base-classes and the specialisations developed by the modeller. The model is constructed at run-time by the designer. The modeller may define methods which derive an attribute value from other attribute values. These methods may be rule-based or they may contain parametric expressions. The designer may use these methods to define dependencies between attributes.

Met : *The methods were implemented as "black-boxes"- a method can return any value which can be computed using the C++ programming language.*

A.5.2 A single attribute value may be derived by several different rules or parametric expressions, typically yielding results of increasing certitude. As the design evolves, the information required for these derivations will gradually become available. When the net is evaluated, the system will automatically "fire" the most suitable derivation depending upon the availability (and possibly also the certainty) of the input attribute values.

Met : *Automatic choice of derivation route based on certainty of input attribute values was not implemented.*

A.5.3 The choice of a suitable derivation may also depend upon the "cost" (in terms of computing time) of each derivation route.

Not implemented :

A.5.4 A single attribute value may also be derived from several rules or parametric expressions simultaneously - each one may be regarded as a piece of evidence concerning the likely value of the attribute. For example, there may be estimates derived from the historical evidence tool and also from the current design. The system will include a choice of techniques to combine these various pieces of evidence. For example, Bayesian estimation (see pp. 97-100 [Beck 1977]) will be used to update the parameters of an analytical PDF using a histogram or a sample from the historical database. If the various methods for deriving an attribute value give widely different or conflicting results, then a warning will be generated.

Abandoned : *Following discussions with the industrial partners, it was concluded that combining evidence in this way would give rise to results which were difficult to audit or justify. The importance of traceability of results was emphasised in these discussions.*

A.5.5 It is anticipated that the derivation of attributes described above will be implemented as follows:
The objects are optionally organised into a *containment hierarchy* (this hierarchy is intended to enable partial evaluations, see A.7 Evaluating the Network below). Each object has a layer number associated with it, which defines its vertical position in the hierarchy. An object may also have a list *of contained objects* and a single *container object.*

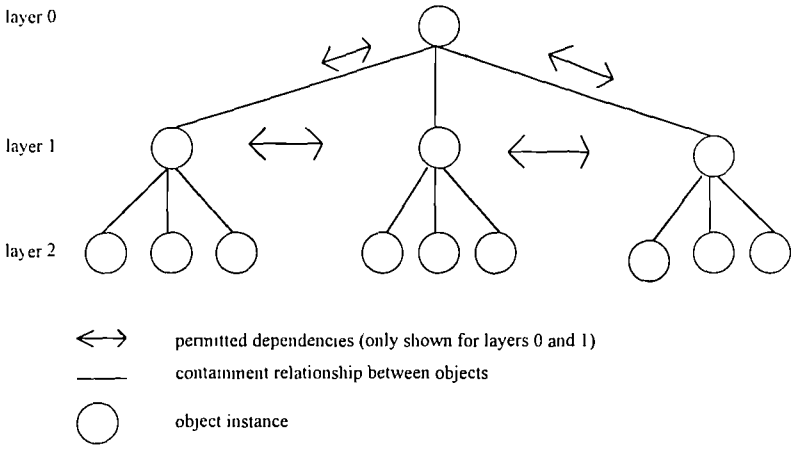


Figure A-1: Containment Hierarchy

There are three ways in which an attribute value can be defined by the designer:

- An attribute of an object may be derived from the attributes of its container, any of its contained objects or any of its peers who share the same container (see Figure A-1).
- The attributes of an object may be derived from the result of a method.
- An uncertain or certain value may be entered for an attribute.

Each attribute may have several different derivations defined, for any mixture of the three derivation routes given above. Each derivation route is assigned a priority which is an integer value indicating whether the results from the different routes should be combined or whether the routes should be used sequentially, as the information becomes available. The currently active priority is the highest priority for which a derivation route has information available. If there are several derivation routes all with the same, currently active, priority, then the results from the different routes will be combined. Various methods for combining results will be provided.

Met / Abandoned / Transformed : *The requirement to combine evidence from different routes concerning a single attribute was abandoned. The requirement to allow attributes to derive their values from arbitrary objects within a containment layer was transformed; it was concluded that it would be very difficult to maintain the integrity of the model if such arbitrary derivation routes were permitted. The benefits of encapsulation would be lost. Instead the usual O-O integrity-maintenance mechanism was adopted, where any sequence of links may be followed, but the link classes must be given by the class definition.*

A.5.6 [CEG] For each simulation, at each level in the containment hierarchy the system will produce an "audit trail", showing which derivation route was used for each attribute. This feature is necessary to provide traceability, given the automatic selection of derivation routes according to what information is available in the model. This "audit trail" may be presented to the user as a text document.

Met :

A.5.7 The audit trail will highlight those cases where the derivation route has changed since the last simulation. It will also highlight those cases where the results from several different routes are being combined.

Not Implemented / Abandoned : *Highlighting changes in derivation route was not implemented. Combination of evidence was abandoned.*

A.6 Alternatives and Scenarios

A.6.1 The designer will be able to develop several alternatives - either attribute values or whole collections of design objects (for example whole branches of a BOM hierarchy). The designer will be able to perform a single simulation which combines all of the alternatives, using assigned probabilities for each alternative (the user will assign these probabilities). He will also be able to select point values for a set of attributes or objects, defining a "what-if scenario", and the risk tool will repeat the simulation for each scenario.

Met / Not Implemented : *Explicit support for "what-if scenarios" was not implemented, but the basic functionality is provided. The user must store a separate file for each scenario, load all the scenario files simultaneously and simulate each file.*

A.6.2 The results of the risk assessment will allow a comparison of the different scenarios selected - for example producing an estimate of the expected cost and the uncertainty for each alternative.

Met : A *"multiple document interface" was implemented, allowing the results obtained from multiple files to be viewed simultaneously.*

A.6.3 If the user performs simulations for several different scenarios, and they each yield different uncertainties for the output attribute, then the system will be able to determine which attribute values in the model made the major contribution to the divergence in output attribute uncertainty.

Not Implemented :

A.7 Evaluating the Network

A.7.1 The system will track changes made by the designer through the dependencies and will flag when data becomes obsolete and a re-evaluation is required.

Not Implemented : *This requirement was not implemented because the model editor was not implemented.*

A.7.2 The system must select the most computationally efficient method to propagate the uncertainties through the network dependencies. For example, analytical techniques would be appropriate when computing the PDF of a linear combination of Normal PDFs. On the other hand, if the dependency is non-linear or the parameters of the original PDFs are not known (e.g. they are histograms derived from historical evidence) then simulation will be used.

Abandoned : *The computational burden of using simulation in all cases was found to be acceptable. The design decision to adopt "black-box" methods resulted in abandoning this requirement.*

A.7.3 In order to render the process of performing a simulation more time-efficient, an implementation capable of performing partial simulations is proposed. Thus, a sub-assembly may be simulated without the whole of the rest of the assembly. If the entire assembly is later simulated, then the results of the previous simulation of the sub-assembly will be re-used, preventing the need for its components to be simulated again.

Not Implemented : *The risk model and tool were implemented in such a way that the storage and retrieval of partial simulation results is possible, but this feature was not implemented.*

A.7.4 Each object has a layer number associated with it, which defines its vertical position in a containment hierarchy. An object may also have a list of *contained objects* and a single *container object*. The attribute values for an object may be derived from either its container object, or from its peers within the same container object or from any of its contained objects. By allowing an object to derive its attribute value from its container object, we provide a mechanism for attributes such as colour to be represented.

Transformed : *The requirement to derive attribute values from arbitrary objects within the containment layer was transformed; attribute values may be derived from any object which can be reached by following links (see 5.5 above).*

A.7.5 The designer will be able to choose the depth of a simulation. Thus a "what-if scenario", defined by the user before a simulation is performed, will include the simulation depth as well as point values for alternatives. The depth of simulation chosen will affect which derivation rules are used - a shallow simulation will tend to use heuristic rules for cost whereas a deeper simulation will use aggregation, for example.

Not Implemented : *Explicit support for "what-if scenarios" was not implemented. Neither was a choice of simulation depths for a given risk model - the simulation depth is always the maximum supported by the risk model.*

A.7.6 The system will be able to use the what-if scenario definitions to piece together existing simulation results and re-use them in a new simulation where possible.

Not Implemented :

A.7.7 When performing a simulation, the system will automatically determine the number of simulation runs which are required to achieve the required accuracy in the output attribute results. The user will be able to specify this required accuracy.

Not Implemented :

A.7.8 If the user has requested a sensitivity analysis for a particular output attribute, the system will use multiple simulations to determine which of the attributes in the simulation are making the major contribution to the uncertainty in the output attribute.

Met /Not Implemented : *Algorithms were developed for performing risk sensitivity analysis, most of which were incorporated into the risk tool. However, some parts of the algorithms were not coded and the necessary modifications to the user interface were not implemented.*

A.8 Historical Data

A.8.1 The designer will be able to find the single nearest-neighbour to a current design or component from the database in order to provide a point value estimate of cost for example. The metric used to define "nearest-neighbour" will be user-definable.

Not Implemented : *A separate historical evidence tool, which makes use of a historical archive of object instances (the database referred to above), was not implemented. A selection of metric methods could however be written on classes to implement this requirement within the existing framework.*

A.8.2 The designer will be able to select a set of previous components from the database, select the major cost-drivers from amongst the attributes (for example using principle component analysis and correlation techniques to help identify them) and then use multiple-variable regression to estimate the cost for the current design from its attribute values. The regression formula thus obtained will be stored in the database for efficient re-use at a later date. The regression formula will be expressed as a hierarchy of equations - the top equation depending only on the major cost-driver, the next equation depending on the top two cost-drivers and so on. Thus the estimate can be refined as more of the cost driving attributes become available.

Not Implemented : *Such a regression formula could be stored by defining it as a method on a class. Automatic generation of such methods from a historical archive was not implemented. Automatic generation of the coefficients for a regression formula from a list of object instances could be implemented, again, using a method defined on a class.*

A.8.3 The designer will also be provided with tools to estimate the parameters of an analytical PDF of known form (for example a beta distribution) from a sample of values, where the sample is taken from the historical database. For example, the method of moments might be used.

Not Implemented : *Could be implemented using a method defined on a class.*

A.8.4 The historical evidence tool will also permit the parameters of a previously estimated analytical distribution to be updated from a sample taken from the historical database (for example using Bayesian estimation). This will be one of the techniques used for "combining evidence".

Not Implemented :

A.9 Graphical and Tabular Output from Risk Assessment Tool

A.9.1 The system will provide various graphs and tables for the risk assessor to use. These graphs and tables may be obtained at any time and at any level in the containment hierarchy. If the simulation results requested are not available, because the model has been modified since the last simulation was performed, then an indicative point value will be provided instead. The risk assessor must decide on the attributes of interest (i.e. the output attribute s) and the "what-if scenarios" before the net is evaluated.

Met :

Graphs and tables which will be provided include:

A.9.2 Table of probabilities or Boolean values for all the individual attribute constraints in the model or in a part of the model.

Met / Transformed : *The requirement to support general modelling of constraints was transformed into the simpler requirement to model goal (or "target") values for attributes. The requirement to show the probability of an attribute value attaining its goal was met.*

A.9.3 [CEG] A summary table or graph of uncertain leaf attribute values present in the model or in a part of the model. The user selects an attribute name and the expected value and uncertainty for all instances of that attribute name in the model will be presented as a *table* or as a *population graph.*

For example, the class `TandemColdMillFunction` might have an attribute `cost_to supply_as_is`. There may be many instances of this class (or classes derived from it) in the model. A summary table for `cost_to_supply_as_is` might look like this:

TandemColdMillFunction . cost_to_supply_as_is

Object Name	Expected Value (£)	Uncertainty (%)	Uncertainty (£)
setup_and adaption	20000	25	5000
process control	10000	40	4000
actuator_referencing_and_sequencing	10000	10	1000
closed loop_control	5000	18	900
actuator interfaces	4000	30	1200
and so on.....	...	...	...

An indication of the "risk profile" for the tandem cold mill project might be obtained by dividing the range of uncertainty values into bands and plotting a population graph, showing the number of objects whose uncertainty lies within each band.

Not Implemented :

A.9.4 Risk simulation results for a single attribute:

A.9.4.1 If the results of the previous simulation are no longer valid (because of changes to the model) then up-to-date data for a plot of probability VS attribute value will not be available. In this case a single indicative point value (the expected value or the most likely value) will be displayed. These indicative point values are re-calculated immediately, each time the model is edited.

Met :

A.9.4.2 There will be a choice of graph types, all representing probability VS attribute value, including *histogram*, *cumulative probability graph* and *statistical tables.*

Met :

A.9.4.3 If there are a choice of simulation results in the database for the chosen attribute (from different what-ifs and different depths of simulation) then they will be presented as a pick-list of scenarios. If the user selects more than one scenario then the choice of graph-type will include *expected value with confidence bands VS scenario* and *statistical tables.*

Not Implemented : *Explicit support for "what-if scenarios" was not implemented, thus results from different scenarios cannot be presented on a single graph.*

A.9.5 Risk simulation results for a pair of attributes (for example cost and expected lifetime for the same assembly):

A.9.5.1 Two graphs may be presented side-by-side, each representing probability VS attribute value with the choice of graph types listed above. The two attribute values may be shown superimposed on the same graph, with two different scales shown on the attribute value axis (the X-axis).

Met / Not Implemented : *Graphs can be shown side-by-side but two attributes cannot be shown on a single graph.*

A.9.5.2 If there is a choice of scenarios and the user selects more than one, then the choice of graph types will include a *"cross" plot* showing, for example, cost VS expected lifetime, with a cross centred on the expected values representing each scenario and the length of the arms of the cross representing the uncertainty in the value.

Not Implemented / Transformed : *Explicit support for "what-if scenarios" was not implemented. A scatter graph can be displayed showing any attribute pair. If the model contains alternatives, all of which are to be simulated, then clusters can often be seen on such a scatter graph corresponding to each alternative, but the requirement to display a "cross" for each alternative was not implemented. The mean, standard deviation etc. for each attribute of each alternative can be displayed textually.*

A.9.6 Sensitivity analysis results for a single attribute:

A.9.6.1 Those uncertain attributes in the model which make the major contribution to the uncertainty in the selected attribute will be identified and ranked. The choice of graph types for the result will include a *pie-chart* where each slice represents the contribution from a particular input attribute to the uncertainty in the output attribute.

Met / Not Implemented : *Textual output of rank and linear correlation coefficients was implemented, providing a way of ranking the numerical sources of uncertainty in the risk model. Algorithms were developed for performing full risk sensitivity analysis on all sources of uncertainty in the model, but graphical output of the results was not implemented.*

A.9.6.2 If there is a choice of scenarios and the user selects more than one, then the choice of graph types will include a *set of pie-charts* (one for each scenario) and a Cartesian plot of *contribution to uncertainty VS scenario*, with a line for each of the major contributors.

Not Implemented : *Explicit support for "what-if scenarios" was not implemented.*

A.9.7 If different scenarios give different uncertainties for an output attribute, the risk assessor may identify where in the model the divergence in output between the different scenarios arose. This is not the same as a sensitivity analysis (above). The divergence may be small compared to the overall uncertainty, so the cause of the divergence may not appear in a sensitivity analysis. The choice of graph types for the result will include a *pie-chart* where each slice represents the contribution from a particular attribute to the scenario divergence.

Not Implemented : *Explicit support for "what-if scenarios" was not implemented.*

A.9.8 Table showing those attributes in the model for which there is significant conflict between the simulation results obtained by alternative derivation methods. The user may select what constitutes a "significant" conflict.

Not Implemented :

A.10 Browsing the Network

A.10.1 [CEG] The designer or risk assessor will be able to browse the various types of entity which comprise the model using indented lists. An indented list may be used to represent any network of nodes connected by directed arcs, although the same node may appear in more than one place in the list. The names of the sub-nodes of a given node (i.e. those nodes which are directly pointed to by an arc originating at the given node) are displayed below the given node in the list and are indented by one place from it. Clicking on a node will display its sub-nodes.

Met :

A.10.2 The types of indented list which may be displayed will include:

Node represents	**Arc represents**	**Arc label (appended to sub-node name in list)**
1.Object	containment	
2.PhysicalObject	IsAComponentOf	cardinality (number of sub-assemblies comprising assembly)
3.Object	IsAlternative	
4.Class	inheritance	
5.Attribute of object	attribute derivation route	method name or other object name AND priority
6.Object or Attribute of class	attribute belongs to object	
7.Attribute of object	attribute interaction constraint	
8.Class	structural selection constraint	a label which indicates whether the creation or deletion of the parent node constrains the deletion or creation of the child node (i.e. one of four cases).
9.Class or Object	object is an instance of class	

Met / Not Implemented : *The requirements to display dynamic indented lists where a node represents an object were implemented (1,2 and 3 above). Because the model editing tool was not implemented, there was no requirement to display either the class inheritance hierarchy (4 above) or the links from class definitions (6 above) dynamically; however the inheritance hierarchy can be viewed graphically (for the base classes) in the on-line Help. The requirement to display the attribute derivation route hierarchy (5 above) was met without using dynamic indented lists. Because the requirements to support general constraint modelling was abandoned, there was no requirement to display such constraints (7 and 8 above). The requirement to display a list of all the objects which belong to a class (9 above) was not implemented.*

A.10.3 The user will be able to provide a filter which will be a text string. Only those nodes whose names contain this string will be displayed in the list.

Not Implemented :

A.10.4 The indented list will be displayed in the left hand side of a split window. The right hand side will display all the entities which are related to the currently highlighted node by any of the types of arc listed in the table above. If the user makes a selection from the right hand window, then the sub-nodes will be presented in the list using the new interpretation of an arc.

Not Implemented :

A.10.5 Where appropriate, nodes will also have suitable display options available to the user from the indented list. For example, "display default value" for an attribute of a class or "display simulation results" for an attribute of an object. Graphs or tables will be generated by these display options where appropriate.

Not Implemented :

A.11 The Modelling Tool

A.11.1 A tool will be provided to assist the modeller in producing new class definitions. This tool will enable the modeller to:

A.11.1.1 Add new specialised classes.
A.11.1.2 Add new attributes and methods to existing class definitions.
A.11.1.3 Modify attributes and methods of existing class definitions.
A.11.1.4 Re-compile the simulation application using the modified class definitions.
A.11.1.5 Modify an existing object model such that it remains consistent with the new class definitions.

Met : *A text-file driven code-generating tool was built to meet this requirement. The code-generating tool does not modify existing object models when the class definitions are changed, but the risk-assessment tool attempts to do so when reading the object model from file. If new classes have been added or new methods, links or attributes have been added to existing classes, there is no difficulty (the new links and attributes are added to existing objects and given a value of UNKNOWN). If the type of an existing attribute or the class of an existing link has been changed, then the old value will not be recognised and a new value of UNKNOWN is assigned. If links or attributes have been removed from a class definition then any existing values are ignored.*

A.12 Editing the Design Model

A.12.1 The design model, the class definitions and the simulation results will all be stored in an object repository.

Met / Not Implemented : *The class definitions are stored in an ASCII text file in a human-readable format. Each design model is stored in a separate ASCII text file, in a human-readable object-based format. The simulation results are not currently stored.*

A.12.2 Owner authority is required for all changes to the model. Both objects and attributes have owners. The object owner also owns all attributes of that object and all objects contained in it.

Not Implemented : *The requirement to provide security features for the risk model was not implemented.*

An attribute or object owner may:

A.12.3 Add a new object (select containment object, class for new object, default attribute values from the schema definition will be used if they exist, user will be prompted for any "dangling dependencies"). Must own container object.

Not Implemented / Met : *The requirement to build a mouse-driven editing tool was not implemented - the user must edit the input ASCII text file.*

A.12.4 Copy an existing object (select containment object, its contained objects will be copied over, the user will be prompted for peer connections). Must own container object.

Not Implemented / Met :

A.12.5 Add a new relationship (for example `is_a_component_of`, `is_alternative_object`, `is_manufactured_on`). Must own container object.

Not Implemented / Met : *The `is_manufactured_on` relationship was not implemented.*

A.12.6 Edit an attribute value. Must own attribute.

The user may enter a value as a point, an interval (for ordered types), a histogram or as a pre-defined probability density function and its parameters.

The user may select a method name from the list of existing methods for this class which have the chosen attribute as their *derived attribute.*

The user may select another attribute of another object.

In each of the three cases above, if there is more than one derivation route for the attribute then the user must select a priority for each route. The user must also select a method which will be used to combine the results if there is more than one derivation route with the same priority.

Not Implemented / Met / Abandoned: *The requirement to support combination of evidence from different derivation routes for a single attribute was abandoned (see 5.4 above); thus priority can be indicated simply by ordering within the text file and there is no requirement to choose a combination method.*

A.12.7 Delete an object. Must own object.

Not Implemented / Met :

A.12.8 Delete a relationship. Must own container object OR all objects taking part in the relationship.

Not Implemented / Met :

A.12.9 Whenever the design model is modified, the indicative point values (see 9 Graphical and Tabular Output From the Risk Assessment Tool, above) for those attributes affected either directly or indirectly by the change will be re-calculated.

Not Implemented :

A.12.10 When a new alternative object is added to the model, this will often require that designers of other parts of the model must modify existing attribute values or enter values for new attributes. In this case, the tool will inform the designers which new information it requires in order to perform a simulation. For example suppose the model of the facia includes a `FaciaInnerMoulding` without a passenger-side airbag (`facia_inner`) and a new `FaciaInnerMoulding` with an airbag (`facia_inner_airbag`) is added as an alternative, for the purposes of a "what-if" analysis. The addition of the airbag will affect not only the design of the inner moulding, but also the design of the face-level `air_vents`. Before a simulation of the facia can be performed, the designer of the air-vents will be informed that he needs to create an alternative `air_vents` object and to enter values for (at least some of) its attributes.

Not Implemented :

A.12.11 It is anticipated that the need to create the alternative `air_vents` object in the example above will be signalled by "dangling dependencies" when the `facia_inner_airbag` is instantiated and the need to enter (some of) its attribute values will be indicated in the "audit-trail" (see paragraph 5.6) which the toolkit produces when a simulation is attempted.

Not Implemented : *The audit trail is only generated for successful simulations.*

A.12.12 A list of missing attribute values will be maintained at each level in the containment hierarchy. This feature will enable the user to ensure that the model is reasonably complete before attempting a simulation.

Not Implemented :

A.13 Discussion

Ideas which have been discussed in connection with the risk assessment tool but which have been omitted from this requirement specification include:

A.13.1 Provision of explicit decision support tools, for example based on a ranking technique. The choice between alternatives will be left to the risk-assessor, although he is free to define a multi-attribute utility function as a derived attribute and thus rank alternatives. Indeed, such a utility function may perhaps be built-in to the base classes supplied with the system.

Not Implemented : *Utility functions were not incorporated into the base classes, but the modeller could easily do so if this were to become a requirement.*

A.13.2 Use of fuzzy numbers as attribute values. This was not felt to be appropriate because, as mentioned earlier, risk assessment concerns the likelihood of an event occurring and the fuzzy calculus does not correctly predict likelihood.

Abandoned :

A.13.3 Explicit support for uncertain FMEA - where the likelihood of a failure, the likelihood of its detection and the consequences of failure are all uncertain. Such an analysis could be performed using the system as specified, if the risk-assessor or designer defines suitable attributes, alternatives and dependencies in the model. Explicit support for "uncertain probabilities" (e.g. using probability intervals) or of FMEA trees could be considered as a requirement in the future.

Not Implemented :

A.13.4 Explicit support for variants and measures of design complexity could be considered in the future.

Met / Not Implemented: *During the prototype tool evaluation at Rover, a clear requirement arose to support modelling of variants and "configuration design". Variants are alternative components and assemblies, only one of which is included in a particular product offering, but all of which will be manufactured - the customer will choose between the offered products. Complexity arises because there may be many different product offerings which can be built by configuring a set of component and assemblies in different ways. Explicit measures of design complexity were not implemented.*

A.13.5 Use of a standard project risk assessment methodology such as RISKMAN. It is desirable that the risk toolkit should be capable of supporting such a methodology, although support for network analysis of project plans (e.g. critical path analysis and PERT) is beyond the scope of this project.

Met / Not Implemented : *The risk tool and risk model are capable of supporting standard risk management practice in the sense that they provide a suitable vehicle for the main key components of such a methodology. However, the tool does not prescribe such a risk management process.*

A.13.6 Fuzzy constraints. It was considered that including both fuzzy and probabilistic representations in the same model would result in an unnecessarily complex model which would be difficult for the designer to build and also for the risk assessor to evaluate.

Abandoned :

Appendix B: Risk and Uncertainty in Engineering Design Models

Contents

B.1 Introduction

This Appendix reviews the uncertainty modelling techniques currently in use in engineering in general and engineering design in particular. The available representations for uncertain information and the available techniques for manipulating this information are categorised. A table is presented in which each technique covered is identified by its data representation, its technique category and also its application areas. It is intended that the remainder of the Appendix, which contains an overview of many of the techniques along with examples of engineering application areas, should be treated as a reference, indexed by this table. Those techniques which are particularly relevent to the aims of this research have been selected from the more exhaustive set given in the indexing table.

In building a design model which incorporates uncertainty, there are three basic issues. Firstly, how to structure the model. Secondly, how to represent an uncertain piece of information in the model. Thirdly, which techniques to use to manipulate the information within the model. Clearly, the structure of the model is strongly related to the techniques which will be used to manipulate the information. The basic data representations for uncertain information considered in this Appendix are:

- sets of point values
- intervals
- probability theory
- fuzzy sets
- mass assignments and pairs of measures.

The techniques reviewed in this Appendix broadly fall into three categories:

- techniques for *propagating uncertainty*
- techniques for finding the *closest match* between objects with uncertain attributes
- techniques for selecting the *best alternative* amongst several possibilities

The categories are not mutually exclusive - some techniques fall into more than one category because they perform more than one function and for others it is ambiguous into which category they should be placed. Nevertheless, these categories provide a useful taxonomy for the techniques.

Techniques for *propagating uncertainty* is a very broad category. It includes techniques for combining several sources of information or pieces of evidence, since this consists of propagating uncertainty from the evidence to the conclusion. It also includes truth-maintenance techniques for evaluating whether or not a set of constraints have been satisfied, since this involves propagating uncertainty from the design and (possibly) the constraint definitions to the conclusion.

Best alternative and *closest match* techniques both comprise methods for search and for evaluation - evaluating the utility for best alternative and the closeness for closest match. Techniques for selecting the *best alternative* usually include sensitivity and uncertainty analysis - analysing the effect of the value and the uncertainty of each input parameter on the value and the uncertainty of the output parameter/s.

The indexing table lists the representations covered in this Appendix and many of the techniques currently used for uncertainty modelling. The position of each technique in the table indicates which data representation (or "approach") it is used with. Below the name of each technique, a letter indicates which of the three categories identified above it falls into:

P: propagation
C: closest match
B: best alternative

Those technqiues which were considered most relevant to the research are described and references are given in Section B.3 "Techniques and Applications". For those techniques which are less relevant there is no further description given in this Appendix, and the table provides references to other work rather than a section within the Appendix.

B.2 Indexing Table

Approach (data represent--ation)	**Technique** (**P**ropagation, **C**losest match, **B**est alternative)	**Application Areas**	**Refs**
Point values	Principal component analysis **C**	Comparing sets of design attribute values.	[Jokinen 1994] [Mardia 1979]
	Taguchi's method **B**	Evaluating "total quality" of a design. Choosing optimal design parameter values.	Section B.3.3.3
	Genetic search as a retrieval technique **C**	Retrieval of component data from databases.	Section B.3.2.1
	Deterministic design evaluation and optimisation **B**	Maximising design objective subject to constraints on parameters. AI systems for automated parameter design.	Section B.3.3.1
Interval	Interval analysis **P**	Rounding errors in computation. Propagating inaccuracies in measurement instruments. Tolerancing. Constraint propagation in automated design.	Section B.3.1.1
	Qualitative geometric modelling **P**	Expressing constraints in conceptual building design.	[Wong 1993]
Probability Theory	Fault tree and failure mode and effect analysis **P**	Engineering reliability.	Section B.3.1.2
	Decision support **B**	Choosing between alternative designs.	Section B.3.3.4
	HAZOP studies **P**	Reliability.	[Andrews 1993]
	Generating PDFs (by parameter estimation, method of maximum entropy and ranks) **P**	Representing sample data and engineering judgement in a probabilistic model. Tolerance analysis/synthesis.	Section B.3.1.3
	Simulation (e.g. Monte Carlo) **P**	Cost models. Tolerance analysis/synthesis. Reliability.	Section B.3.1.4
	Bayesian belief networks **P**	Expert systems (e.g. for medical applications). Process control.	[Pearl 1987] [Kwaan 1994] [Dempster 1994]
	Markov networks **P**	AI Reliability networks	[Andrews 1993]
	Probabilistic retrieval **C**	Retrieval of documents from databases Relevance feedback	Section B.3.2.2

Indexing Table (part 1)

Approach (data represent-ation)	**Technique** (**P**ropagation, **C**losest match, **B**est alternative)	**Application Areas**	**Refs**
Probability Theory	Learning methods for retrieval **C**	Retrieval of documents from databases	[Fuhr 1994] [Savoy 1994]
	Level 1,2 and 3 reliability methods **P**	Structural reliability	Section B.3.1.7
	Probabilistic design evaluation and optimisation **B**	Maximising design objective subject to constraints on parameters	Section B.3.3.2
	Parametric cost estimation **P**	Estimating component costs in the conceptual design phase	[Mileham 1993]
	Tolerance analysis methods **P**	Tolerance design in engineering design	Section B.3.1.8
	Numerical probablistic methods (numerical convolution, DPDs, histogram interval-based methods) **P**	Tolerance analysis/synthesis Reliability modelling Engineering project time/cost modelling	Section B.3.1.5
	Safety Factors **P**	Mitigating against effects of uncertainty in engineering design	Section B.3.1.6
Fuzzy sets	Extension principle **P**	Estimating system uncertainty from parameter uncertainties Fatigue predictions Civil engineering	Section B.3.1.9
	Fuzzy optimisation **B**	Maximising design objective subject to constraints on parameters	[Díaz 1989] [Otto 1991]
	Fuzzy database storage/retrieval **C**	Retrieval of component data from databases	Section B.3.2.3
	Heuristic search **B**	AI for design automation	pp. 181-216 [Dubois 1986]
	Retrieval by fuzzy neuro-computing **C**	Design retrieval	[Bahrami 1992]
Mass assignments and pairs of measures	Dempster-Shafer evidence theory **P**	Combining conflicting evidence	Section B.3.1.10
	Support logic (approximate reasoning in expert systems) **P**	Expert systems for managing risk: Design of aircraft components Modelling uncertainty in nuclear waste management	Section B.3.1.11
	Interval Probability Theory **P**	Civil engineering	Section B.3.1.12

Indexing Table (part 2)

B.3 Techniques and Applications

B.3.1 Propagation Techniques

In this section we give an overview of some of the techniques which may be used to propagate uncertainty from a set of input parameters to a set of output parameters. The relationship between the input and output parameters is part of the model. If the parameter values are intervals, random variables or fuzzy numbers, then the inputs and the outputs may be related by a function, thus we include some techniques for evaluating a function on such parameters: **interval analysis, simulation, numerical probabilistic methods, level 1, 2 and 3 reliability methods, extension principle** and **tolerance analysis methods.** In **fault tree and failure mode and effect analysis**, the function relating the inputs and the outputs is represented as a network.

Support logic (which uses **Dempster-Shafer evidence theory)** and **interval probability theory** are techniques for using logical inference (or induction) to propagate the uncertainty in the available evidence in order to evaluate the evidence in support of a conjecture. Here the parameters could be regarded as uncertain logical variables.

In addition to uncertainty in the parameter values, there may also be uncertainty in the relation between the inputs and the outputs. Thus we must consider both deterministic and uncertain propagation. The propagation of random variables through a function provides an example of deterministic propagation whereas inference from fuzzy propositions using uncertain rules in support logic is an example of uncertain propagation. The **safety factors** often used in engineering represent uncertainty both in the inputs and in their relationship to the outputs.

The determination of the equation and parameters for PDFs from sampled values in **generation of PDFs** can also be regarded as a technique for propagating uncertainty. In this case, we are propagating the uncertainty which is due to the diversity of the observations.

B.3.1.1 Interval Analysis

An interval may be thought of as a new kind of number and interval arithmetic may be used to compute functions of such numbers. Let

I[Xl, Xh] represent the interval $X = \{x : x >= Xl, x <= Xh\}$,

[Yl, Yh] represent the interval $Y = \{y : y >= Yl, y <= Yh\}$

and

[Zl, Zh] represent the interval $Z = \{z : z >= Zl, z <= Zh\}$

Then the rules for interval arithmetic (see [Moore 1966]) are given by:

$$X + Y = [Xl + Yl, Xh + Yh]$$

$$X - Y = [Xl - Yl, Xh - Yh]$$

$$X \times Y = [\min(Xl * Yl, Xh * Yh, Xl * Yh, Xh * Yl), \max(Xl * Yl, Xh * Yh, Xl * Yh, Xh * Yl)]$$

$$X / Y = [[Xl, Xh] \times [1 / Yl, 1 / Yb] \text{ if } 0 \notin [Yl, Yh]$$

Multiplication and addition are both commutative and associative, as with ordinary arithmetic. However, unlike ordinary arithmetic, the distributive law does not always hold. That is:

$$X \times (Y + Z) \neq (X \times Y) + (X \times Z)$$

We do however have the following property:

$$X \times (Y + Z) \subseteq (X \times Y) + (X \times Z)$$

which is known as **subdistributivity** (see [Moore 1979, p. 13]). In [Choobineh 1992] the author considers propagating uncertainty represented by intervals through cost equations and points out that the lack of the distributive property means that the width of the resulting interval depends upon how the equation is expressed. This is clearly unacceptable.

The reason for the subdistributivity problem is that the same interval (X) occurs twice on the left hand side of the equation. If (X x Y) is evaluated, then (X x Z) and then the two results are added then the result may be wider than it should be because the two instances of X are treated as independent. Thus for example two results obtained using the largest value in X to evaluate (X x Y) and using the smallest value in X to evaluate (X x Z) may be added. This incorrect since the variable can only take one value at a time.

Choobineh then presents the **vertex method** (originated by Dong and Shah in [Dong 1987a]) as a technique to avoid subdistributivity. The vertex method is described below.

Suppose f is a function of n interval variables:

$$Y = f(X1, X2, ..., Xn)$$

where

[Xli, Xhi] represents the interval Xi = {x : x >= li, x <= hi}

The n intervals may be represented by an n-dimensional rectangular region. The vertices consist of all the points whose co-ordinates are one of the two limits of each of the n intervals. Thus the number of vertices, Q, is equal to 2^n. Let the vertices be

v_j where j = 1,...,Q

If f is continuous and there are no extreme points of f in the rectangular region or on its boundaries (i.e. f is monotonic within the rectangular region) then the interval Y is given by

$$Y = \left[\min_j(f(v_j)),\ \max_j (f(v_j)) \right] \quad \text{for } j = 1,...,Q$$

If the function f has k extreme points $e_1,...,e_k$ within the rectangular region then Y is given by

$$Y = \left[\min_{j,k}(f(v_j), f(e_k)),\ \max_{j,k} (f(v_j), f(e_k)) \right] \quad \text{for } j = 1,...,Q \text{ and } k = 1,...,K$$

Although more computationally complex than standard interval arithmetic, the vertex method satisfies the distributive law, so that the order in which an expression is evaluated no longer affects the width of the resulting interval.

An application for interval analysis in Engineering Design is in tolerance accumulation. In [Chase 1991], the author presents two models, the "worst case" and the "root sum squared". If the "worst case" assumption is made, then the tolerances of n individual components are aggregated using the equation

$$dU = \sum_i (| df / dx_i | T_i) \quad \text{for } i = 1,...,n$$

where dU represents the variation of the dimension U of the assembly, the T_i represent the component tolerances and f(x1,..., xn) is the assembly function relating U to the component dimensions x_i. Thus df/dx_i is the sensitivity of U to the tolerance of component i. So for a one dimensional assembly, $|df / dx_i| = 1$.

It can be seen that in this model, the T_i may be regarded as the widths of intervals - i.e. the tolerance of a component is represented by an interval. The subdistributivity property of standard interval arithmetic doesn't pose a problem in this context because only linear combinations of intervals are calculated.

A more recent application of interval analysis in engineering design is Ward and Seering's mechanical design "compiler" [Ward 1993a][Ward 1993b], a computer system which automatically selects components from a pre-defined catalogue to optimally implement a design specified by the user. The design is specified as a schematic, a specification and a utility function. What is unusual about Ward and Seering's approach is that,

rather than enumerating all the possible combinations of available components in a conventional optimisation, they represent the component catalogue information and the specified design using intervals. They have developed an inference system based on a calculus of "labelled intervals" which the computer system uses to deduce the optimal design. A labelled interval describes a set of catalogue components and consists of an interval (a pair of real numbers), a variable name and a label which effectively describes the meaning of the uncertainty and thus enables inferences to be made. A variable is identified as being either a *parameter*, which is fixed at manufacturing, such as a gear ratio for a motor, or a *state variable*, such as torque, which may vary during operation. Implicit equations are also attached to sets of components, relating their state variables and parameters (these must be invertible algebraic expressions of the form $f(x,y,x) = 0$, and the calculus also appears to rely on the monotonicity of the function f with respect to each variable). There are six labels, or types of interval:

R[] := *Required, only*
A[] := *Assured, only*
N[] := *No-stronger, only*
R↔ := *Required, every*
A ↔ := *Assured, every*
N ↔ := *No-stronger, every*

The meanings are as follows:

[] *Only*
: The interval represents the limits of the variable. This indicates that values of the variable will or must be drawn *only* from the interval. Examples would be tolerances, or limits which are specified to avoid causing damage.

↔ *Every*
: The interval represents the operating region of the variable. This indicates that the variable will or must assume *every* value in the interval. Examples would include ranges of operating conditions which may occur when the design is in use.

R *Required*
: This indicates that the variable must take a value within the interval if the artefact is to function correctly.

A *Assured*
: This indicates that the variable is guaranteed to only take values within the interval for every component in the set.

N *No-stronger*
: This indicates that there is no subset of the set of the components which can be guaranteed to have stronger limits on the variable.

The following five standard interval operators are used, each of which takes a pair of intervals as its argument (an example is given for a variable *x)*:

$\cap((x\ 1\ 4),(x\ 2\ 6)) \rightarrow (x\ 2\ 4)$
$\neg \cap((x\ 1\ 4),(x\ 2\ 6)) \rightarrow$ FALSE
$\cup((x\ 1\ 4),(x\ 8\ 10)) \rightarrow (x\ 1\ 10)$
$\subseteq((x\ 10\ 12),(x\ 10\ 14)) \rightarrow$ TRUE
$\neg \subseteq((x\ 10\ 12),(x\ 10\ 14)) \rightarrow$ FALSE

Three new operators are also defined which take an implicit *equation* in three variables and a pair of intervals in two of the variables (say x and y) as their argument, and return an interval in the third variable (say z). These are RANGE, DOMAIN and SUFPT ("sufficient points"). RANGE is used to determine the range of values which is taken by z when x and y vary independently over all the values in their intervals - as defined in conventional interval arithmetic. This represents the situation where the designer is free to choose any one value within its interval for x and any one value within its interval for y, and wants to know what range of values z may therefore take - i.e. traditional constraint propagation. DOMAIN is used to determine the range of values <u>all</u> of which z must be able to take if x is to be able to take all values in its interval and y may take any value in its interval. DOMAIN is the inverse of RANGE:

$$\text{DOMAIN}\big(f(x,y,z)=0,\ (x\ X_{lo}\ X_{hi}),\ (y\ Y_{lo}\ Y_{hi})\big) \rightarrow (z\ Z_{lo}\ Z_{hi})$$
$$\Leftrightarrow$$
$$\text{RANGE}\big(f(x,y,z)=0,\ (z\ Z_{lo}\ Z_{hi}),\ (y\ Y_{lo}\ Y_{hi})\big) \rightarrow (x\ X_{lo}\ X_{hi})$$

SUFPT is used to determine the range of values any one of which may be taken by z if x must take at least all values in its interval when y varies over all the values in its interval. This is a different kind of inverse to RANGE:

$$\text{SUFPT}\big(f(x,y,z)=0,\ (x\ X_{lo}\ X_{hi}),\ (y\ Y_{lo}\ Y_{hi})\big) \rightarrow (z\ Z_{lo}\ Z_{hi})$$
$$\Leftrightarrow$$
$$\forall Z_{fix} \in [Z_{lo}\ Z_{hi}]$$
$$\text{RANGE}\big(f(x,y,z)=0,\ (y\ Y_{lo}\ Y_{hi}),\ (z\ Z_{fix}\ Z_{fix})\big) \rightarrow (x\ X_{lo2}\ X_{hi2})$$
$$\text{where } [X_{lo2}\ X_{hi2}] \supseteq [X_{lo}\ X_{hi}]$$

Using these eight basic operators as building blocks, Ward and Seering are then able to define rules for performing three activities on labelled intervals; *elimination*, *abstraction* and *propagating labelled intervals using equations*. Elimination operations are used to remove sets of artefacts which conflict with specifications imposed by the user or by other parts of the design. Abstraction operations are used to determine a labelled interval for the union of some sets of artefacts from the labelled intervals of the constituent sets. By propagating labelled intervals through an equation, the design compiler is able to determine the labelled intervals for the particular variables required for the other operations. The rules effectively specify the basic operators to be used for each type of interval, or for each combination of types of interval.

In this way, the design compiler is able to reduce the complete sets of options for each component provided in the catalogue to smaller sets which satisfy the constraints specified by the user in the labelled interval specification language. A simple utility function (e.g. cost + weight) is then used to select the optimum design. This approach is more computationally efficient than enumeration and evaluation of all combinations of components .

B.3.1.2 Fault Tree and Failure Mode and Effect Analysis

Fault tree and failure mode and effect analyses are two important techniques in safety analysis, used to evaluate the probability of an undesirable event.

In a fault tree analysis, a causal hierarchy is constructed. The undesirable event is placed as the root node. Then the sub-nodes, consisting of all the possible causal events for the root node are determined. At each node, the causal sub-nodes are combined by gates. Types of gate are shown below (from [Andrews 1993]):

AND gate	output event occurs if all input events occur simultaneously
OR gate	output event occurs if at least one of the input events occurs
m out of n (voting gate)	output event occurs if m out of n input events occur
XOR gate	output event occurs if one, but not both, of the two input events occurs
inhibit gate	input produces output when the input event and the conditional event occur
Priority AND gate	output event occurs if all input events occur in the order from left to right
NOT gate	output event occurs if the input event does not

If the probabilities of the leaf events are not known, a qualitative analysis may still be performed on the fault tree by identifying the minimal cut sets. This is the minimum set of leaf events which will cause the root event - if any one event is removed from a minimal cut set, the root event will not occur. Simple algorithms exist for identifying the minimal cut sets from the fault tree.

If probabilities are assigned to each leaf-node then the exact probability of the undesirable event may be calculated directly from the minimal cut sets provided that the nodes are independent. For large trees, this is not practical and so the probability is expressed as a series (called the inclusion-exclusion expansion) which is truncated to obtain an approximation to the failure probability. By truncating at even and odd terms, an upper and lower bound for the probability may be obtained.

Thus fault tree analysis is a "top-down" approach. In contrast, failure mode and effect analysis takes a "bottom-up" view of safety analysis. The system is broken down into components or sub-assembly blocks and each block is examined for its "modes of failure". Each mode of failure is then classified according to the

effect it has on the system. Each component failure mode is then given a rank from, say, 1 to 10, which reflects its likelihood and another which reflects the risk of it not being detected. The consequences of failure are ranked according to dis-utility. By multiplying these ranks together, a risk ranking can be obtained for each event and action can then be taken to prevent those events with a high risk ranking.

There is a choice between the functional (or black box) approach where the leaf nodes are whole sub-assemblies and the hardware approach where each component of each sub-assembly is considered.

B.3.1.3 Generation of PDFs

An important set of techniques in probability theory are those required to produce an equation for a probability density function from a sample of values taken by a random variable. Three methods will be discussed here:- parameter estimation (pp. 84-87 [Siddall 1983]), maximum entropy (pp. 92-126 [Siddall 1983]) and using ranks (pp. 87-92 [Siddall 1983]).

parameter estimation

The simplest method to estimate the parameters of a known distribution function from a sample is to estimate the moments of the distribution to be equal to the moments of the sample. This is known as the method of moments. For example, if a sample $x_1,...,x_n$ is known to fit a normal distribution then the first two moments are μ and σ^2 and can be estimated by

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

and

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - \mu)^2$$

A second method is known as the method of maximum likelihood. The method of maximum likelihood is based upon calculating the parameters which are most likely to have produced our sample. For example, if our sample is known to be taken from an exponential distribution (with PDF $f(\lambda)=\lambda e^{-\lambda x}$) then the probability of achieving the sample $(x_1,...,x_n)$ is given by

$$Pr(x_1,...,x_n \mid \lambda) = Pr(x_1 \mid \lambda) * Pr(x_2 \mid \lambda) *...* Pr(x_n \mid \lambda)$$

Thus we wish to maximise the likelihood function L given by

$$L = \lambda e^{-\lambda x_1} \cdot \lambda e^{-\lambda x_2} \cdots \lambda e^{-\lambda x_n}$$

$$= \lambda^n e^{-\lambda \sum_{i=1}^{n} x_i}$$

For convenience, we will actually maximise

$$\ln(L) = n \cdot \ln(\lambda) - \lambda \sum_{i=1}^{n} x_i$$

When L is at a maximum

$$\frac{\partial(\ln L)}{\partial \lambda} = \frac{n}{\lambda} - \sum_{i=1}^{n} x_i = 0$$

Thus

$$\lambda = \frac{n}{\sum_{i=1}^{n} x_i}$$

A third method for parameter estimation is known as Bayesian estimation (see [Beck 1977, pp. 97-100]). Bayes theorem is used to improve upon an initial estimate for the distribution parameters. In Bayesian estimation, the *distribution parameters themselves* are regarded as random variables. A PDF must be provided

for the distribution parameters (usually subjectively estimated) and Bayes' theorem is then used to combine a sample of data with this *a priori* distribution to obtain an improved estimate for the distribution parameters.

For example, suppose we wish to estimate λ for the exponential function as above. We must first make an estimate of a distribution for λ. Suppose our prior estimate for λ is that it lies between 0 and 1 and is approximately 1/2 then we might provide a prior distribution of

$$g(\lambda) = 6(\lambda - \lambda^2)$$

The formulation of Bayes' theorem for a continuous parameter is

$$g(\lambda|x_1,...,x_n) = \frac{g(\lambda)\cdot f(x_1,...,x_n|\lambda)}{\int_{-\infty}^{+\infty} g(u)\cdot f(x_1,...,x_n|u)du}$$

where f(x1,...,xn | λ) is the likelihood function L, defined above. Thus an improved estimate for g(λ) is obtained. An improved estimate for λ could then be obtained by taking the expected value of g(λ).

Siddall raises the objection [Siddall 1983, p. 73] that in Bayesian estimation it is not the primary distribution itself which is treated as subjective. Yet in engineering, the engineer's expertise concerns the primary distribution and this is where his subjective judgement should be directed. Siddall also objects that an illusion of rigor is provided - statisticians generally accept 5% confidence limits on the parameter estimation as grounds for accepting the primary distribution, yet the primary distribution could be based upon entirely the wrong analytical function.

using ranks

If the analytical form of the PDF is known, then is some circumstances its parameters can be arrived at using ranks and order statistics [Siddall 1983, pp. 87-92]. The sample is ranked in order of value from low to high. Thus

O_i = i'th member of the ranked sample

O_i is known as the ith order statistic and is clearly a random variable. If F is the cumulative distribution and f is the PDF of the primary random variable, X, then

$$r_i = F(O_i) = \int_{-\infty}^{O_i} f(x)dx$$

is known as the ith rank. We want to estimate the r_i corresponding to each O_i and we use the mean rank for this purpose. It can be shown that the mean rank is given by

$$\bar{r}_i = \frac{i}{n+1}$$

If the PDF can be linearised by a transformation T, then the same transformation can be applied to the (O_i, r_i) pairs and the best-fit straight line can then be fitted through them.

maximum entropy

We can use the concept of entropy in conjunction with Jaynes' principle to generate a PDF from a sample when we have no prior information about the likely distribution function. Jaynes' principle (page 92 [Siddall 1983]) states that:

> "The minimally prejudiced probability distribution is that which maximises the entropy subject to constraints supplied by the given information."

In his "Algebra of Probable Inference" [Cox 1961, pp. 35-68] Cox characterises the entropy of a set of propositions as a measure of diversity and uncertainty. For an exhaustive set of n equiprobable and mutually exclusive propositions, he defines the entropy to be:

$$\eta(n) = \ln(n).$$

For example, if the propositions are the 52 propositions that a particular card will be drawn from a well-shuffled pack then it is intuitive that the diversity of the choice of card is composed of the diversity of the choice of suit plus the diversity of the choice of card within the suit. This requires that:

$$\eta(52) = \eta(13) + \eta(4)$$

which is satisfied by the use of the logarithm in the definition. Entropy may also be regarded as a measure of the amount of information in a set of propositions - the greater the entropy, the more questions are required to determine which proposition is true.

We now permit the propositions to have different probabilities (as in the case of the set of propositions concerning the value taken by a random variable which are represented by a probability distribution). Thus we require our definition of the entropy of the set of propositions $a_1,...,a_n$ to be such that the entropy is greatest when the probabilities are all equal, since in this case the uncertainty is greatest. We also require that the contribution of an impossible proposition to the entropy should be zero. These requirements are met by the definition:

$$\eta(a_1,\ldots,a_n) = -\sum_{i=1}^{n} Pr(a_i)\cdot \ln(Pr(a_i))$$

Another way to regard the definition of entropy originates in information theory (page 93 [Siddall 1983]). Entropy is regarded as the mean information over all the propositions, where the information contained in a proposition is given by:

$$\eta(a_i)=-\log(Pr(a_i))$$

in order to make the entropy of joint independent events additive. The choice of base for the logarithm is a matter of convention. If we choose to take our logarithms base e, then the definition of the entropy of a continuous PDF is

$$S[f(x)] = -\int_R f(x)\cdot \ln[f(x)]dx$$

If we have no information about the likely analytical form for the PDF, then Jaynes' principle tells us that maximising the entropy will give the least biased form. If we require that the first m moments, $m_1,...,m_m$ of the PDF should match those of the sample then maximising the entropy using Lagrangian multipliers (page 100 [Siddall 1983]) we arrive at the solution

$$f(x) = e^{\left(\lambda_0+\sum_{i=1}^{m}\lambda_i\cdot x^i\right)}$$

which is known as the *maximum entropy density function.* A set of m equations in the m variables λ_i can be arrived at:

$$m_j = \frac{\int_R x^j\cdot e^{\sum_{i=1}^{m}\lambda_i\cdot x^i}\,dx}{\int_R e^{\sum_{i=1}^{m}\lambda_i\cdot x^i}\,dx} \qquad j=1,\ldots,m$$

The values of the λ_i can then be calculated using a non-linear programming technique. A similar method may also be applied using ranks instead of matching the moments.

B.3.1.4 Simulation

There are three basic approaches to combining probability density functions (PDFs) through functions - the analytical, the numerical and simulation. If you have an analytical description of the PDF for one or more random variables it is sometimes possible to calculate the distribution of a resultant random variable analytically (for example using moment-based methods). This is generally only practical for very simple and well-defined situations - such as taking a linear combination of normal distributions - partly because you need to know the equations for the input distributions and the output distribution and partly because for all but the most trivial problems the analysis soon becomes unmanageable. The numerical methods (Section 5) are either based on numerical convolution (which requires that the combining function is known in closed form and possesses a unique differentiable inverse) or on approximating the input PDFs to discrete/histogram forms (which introduces bias).

The third approach is simulation, most commonly Monte Carlo simulation. Simulation is more generally applicable and can ultimately yield more accurate results than numerical approaches, but the computational cost is higher. In a Monte Carlo simulation, random numbers are used to sample from a probability distribution. Monte Carlo simulation is often explained by drawing an analogy between generating a random number and taking a gaming chip from a bag.

Suppose we have N independent discrete random variables, X1,...,XN, we know their distributions and we wish to calculate the distribution of the random variable Y, where Y = f(X1,...,XN). For each random variable X_i, we take say 1000 chips and mark a value on each chip (one of the discrete values taken by the random variable). We use the PDF for X_i to determine how many chips to mark with each value. Then we place all the chips in a bag. Thus we have N bags of chips.

We then select a chip at random from each bag, giving us values for the random variables $(x_1,...,x_n)$. We calculate $f(x_1,...,x_n)$ and store the result. This is repeated 1000 times, giving us a sample of 1000 values for Y which provides an approximation to the PDF for Y. The set of values obtained for the x_i is known as a theoretical sample.

A method which is widely used to generate the theoretical sample in practice is known as the **rejection method** (see [Siddal 1983, pp. 150-176]). The rejection method may be used for continuous random variables. Suppose X is a random variable which may take values between l and u. Let the PDF of X be p(x) and let it have a maximum value of a. Then two random variables, r1 and r2 which lie between 0 and 1 are generated from a uniform distribution. Then a sample value for X is computed

$$x = r_1 * (u - l) + l$$

If

$$r2 <= p(x) / a$$

then the sample value is accepted. Otherwise, it is rejected. One problem with the rejection method is that it is inefficient in that more random numbers are generated than are actually used. In [Deák 1981], a more economical method which makes use of all the values generated is proposed.

The issue of how to produce a PDF for the output variable from the sample is covered in **Generation of PDFs,** above.

If the X_i are dependent, then the sample values are no longer independent. For example, suppose that X_1 and X_2 are dependent. Then their joint distribution could be expressed either as $p(x_1, x_2)$ or as $p(x_1)$ and $p(x_2 | x_1)$. If we know $p(x_1, x_2)$ then we must generate three random variables for the rejection method, r_1, r_2 and r_3. The sample values for X1 and X2 are computed

$$x1 = r_1 * (u1 - l1) + l1$$
$$x2 = r_2 * (u2 - l2) + l2$$

and the decision is made to accept or reject the pair (x1, x2) depending upon whether

$$r3 <= p(x1, x2) / a$$

If we know $p(x_1)$ and $p(x_2 \mid x_1)$ then a sample value for x1 is generated and accepted or rejected as usual. Then a sample value for x_2 is generated and is either accepted or rejected depending upon whether

$$r4 <= p(x2 \mid x1) / a$$

A problem with Monte Carlo simulation is that because the samples are generated randomly, most of the sampled values will come from the centre of the distribution. Thus if only a small number of iterations are performed "clustering" occurs - the values in the outer parts of the variable distributions aren't represented in the samples. This is particularly important if low probability outcomes are of interest.

Alternative sampling methods which overcome the problem of "clustering" and require fewer iterations to reconstruct the output PDF are known as **stratified sampling techniques**. A widely used example is known as Latin Hypercube sampling (see [Iman 1980]).

In Latin Hypercube sampling, the input distribution is stratified - the cumulative curve is divided into equal intervals on the cumulative probability axis. The number of intervals is equal to the number of iterations to be performed. In simple Latin Hypercube sampling one sample is taken from each interval, ensuring that all sections of the cumulative probability distribution get equal representation in the sample. Thus with Latin Hypercube sampling you build up an accurate re-creation of the frequency distribution of the output function much more quickly than with Monte Carlo sampling.

In **midpoint Latin Hypercube sampling** (see [Morgan 1990, pp. 204-205]), the mid-point of each interval is stored for each input parameter. A sample is generated by selecting a value at random for each input parameter. Once a value has been selected, it is removed from the stored list. Thus, the resultant sample is even more uniformly spread than for simple Latin Hypercube sampling.

In [McKay 1979], Monte Carlo sampling is compared with stratified sampling and the Latin Hypercube technique.

There are other criteria than cumulative probability which may be used to determine the sampling strategy. The **directional simulation** method [Proban 1989, pp. 75-77] involves taking samples in directions which are uniformly distributed on the surface of a sphere in the n-dimensional space defined by the n input parameters. The **axis orthogonal simulation** method [Proban 1989, pp. 78-82] is used in reliability analysis.

B.3.1.5 Numerical Probabilistic Methods

Numerical Convolution

This class of methods provides a way to obtain the PDF of an output random variable Y, where Y = f(X1,...,XN) and the distributions of the N input random variables are known. The function f must be known in closed form. For simplicity, here we consider the case where there are only two input variables. Let the PDFs of Y, X1 and X2 be P_Y, P_{X1} and P_{X2} respectively. If the function f possesses a differentiable inverse f^{-1} such that:

$$Y = f(X1, f^{-1}(Y, X1))$$

then the PDF of Y is given by:

$$P_Y(Y) = \int_{-\infty}^{+\infty} P_{X1}(X1) \cdot P_Y(f^{-1}(Y,X1) \mid X1) \cdot \left| \frac{d\, f^{-1}(Y,X1)}{dY} \right| dX1$$

For example, if the function f is simply:

$$Y = f(X1, X2) = X1 + X2$$

then the PDF of Y is given by the convolution integral (see, for example, p. 188 [Davenport 1970]):

$$P_Y(Y) = \int_{-\infty}^{+\infty} P_{X1}(X1) \cdot P_Y(Y - X1)\, dX1$$

Even for the simple additive case above, it is generally extremely difficult to obtain an analytic solution to such integrals, particularly if there are more than two input variables. However, numerical integration techniques can be applied. In [Varghese 1996] for example, the numerical convolution technique is applied

to a simple "tolerancing stack-up" problem, calculating the distribution of the sum of two random variables which represent dimensions of components in an assembly.

Discrete Probability Distributions (DPDs)

This approach is only applicable if the input variables are independent. A probability mass function [Hays 1970] (otherwise known as a discrete probability distribution or DPD [Cooper 1987]) is a simplified representation for a random variable, where only discrete points on the PDF are considered. It consists of a discrete set of <probability, value> pairs, where the sum of the probabilities is normalised to be equal to 1. The DPD of a function F of N independent random variables $X1$,..., XN whose individual PDPs are known can be calculated easily. Considering two variables, for example, if

$$Y = F(X1,X2)$$

and the DPD of $X1$ is

$$\{<Pi1, Xi1>\}\ i=0,..I$$

and the DPD of $X2$ is

$$\{<Pj2, Xj2>\}\ j = 0,...,J$$

then the DPD of Y is

$$\{<Pi1Pj2, F(Xi1, Xj2)>\}\ \ i=0,..I, j = 0,...,J$$

If the output from this algorithm is to be used as input to another evaluation, and the algorithm is to be repeatedly performed, it is clearly necessary to reduce the number of pairs back to some fixed value, M, after each evaluation. This can be achieved by dividing the domain of Y into M equal intervals and combining the N pairs $<Pi, Yi>$ which lie within an interval using some aggregation function for Y such as the mean, to give a single pair $<\Sigma Pi, \Sigma Yi/N>$.

The initial approximation may introduce significant error if the number of discrete values chosen is insufficient to represent the detail of the original PDF. More seriously, in a complex model containing many functions, with the output from one acting as input to the next, repeated reduction back down to M values will introduce a systematic error which will increase as each function is evaluated.

Histogram Representations

An alternative approach is to use a histogram representation for the PDFs of the random variables, as advocated by Chapman and Cooper for their controlled interval and memory (CIM) method (see [Cooper 1987]). The common interval (CI) methods which they propose provide a computationally efficient and arbitrarily accurate means of propagating probabilistic uncertainty through functions. However, the CI approach must be tailored to a given functional form; the combining function must be known in closed analytic form and thus, unlike Monte Carlo simulation, CI methods are not applicable to arbitrary, "black-box" functions.

Suppose that Y = F(X1, X2) where the function F and the distributions of X1 and X2 are known, X1 and Y1 are independent, and we wish to determine the distribution of Y. Taking the CI approach we represent the distributions of X1 and X2 by histograms each with M bars or intervals of equal width (it is not necessary to have the same number of intervals for each input but simplifies the explanation). The intervals divide the domain of each input random variable into a set of M classes and each class has a class marker which is its mid-value. By the principle of indifference, in the absence of any other information concerning the shape of the input distribution within a histogram interval, it must be assumed uniform, yielding a step-wise rectangular distribution.

The simplest but least accurate method is to effectively treat the class markers as though they were the values taken by a discrete probability distribution. Suppose the histogram representing X1 has class markers {x11, x12,..., x1M} with probabilities {p11, p12,..., p1M} respectively and similarly that X2 is defined by {x21,...,x2M} and {p21,...,p2M}. Since X1 and X2 are independent, we can associate a probability of p1i × p2j with the value F(x1i, x2j) for i=1,..,M and j=1,..,M. If the same value occurs more than once in the set of values {F(x1i, x2j)}, then the corresponding probabilities are added. Thus we arrive at a new set of up to M^2 <value, probability> pairs. The simplest method is to interpret these as the class markers and probabilities for a new histogram (as though the histogram were a DPD); but this can introduce significant error into the

results. It would only yield the "correct" values for the output histogram if the distribution of F(X1, X2) were uniform given uniformly distributed inputs. For example, suppose F(X1, X2) = X1 + X2. The distribution of the sum of two variables with uniform distributions of the same width is triangular, not uniform (see Figure B-1). Thus the results obtained by treating the input distributions like DPDs are incorrect, but given knowledge of the form of F (and hence of the correct distribution shape for F evaluated on uniform distributions) they can be corrected. Let g() be the PDF for F evaluated on uniform distributions (e.g. g() is triangular in the additive example).

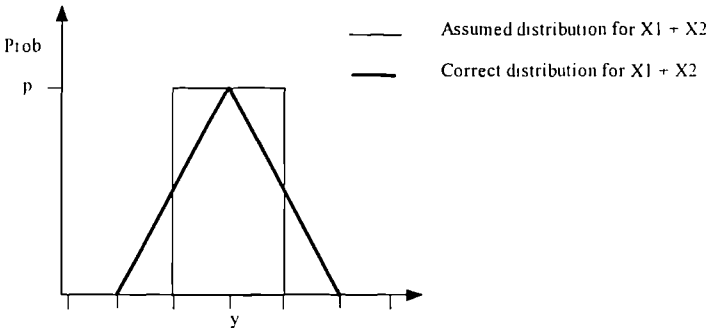


Figure B-1: Distribution of sum of two independendent, uniformly distributed, random variables

The main approach advocated in [Cooper 1987] is to increase the number of intervals in the output histogram, so that there are several (say N) sub-intervals corresponding to each of the initially calculated <value, probability> pairs. A fixed set of factors are calculated which represent the value taken by g() at the centre of each sub-interval. In the additive example there will be 2 × N factors because the "correct distribution" shown in Figure B-1 covers twice the range of the "assumed distribution". Each initial <probability, value> pair is then multiplied by the set of factors to yield 2 × N such pairs, and the probabilities corresponding to matching classes are added as usual. Precomputing a fixed set of factors is a computationally efficient approach and accuracy can be increased to an arbitrary level by increasing N. The approach described yields the same results as numerical functional integration - but the convolution integral has been decomposed in a computationally convenient fashion. The representation is different however, since in functional integration the histogram is an approximation to an underlying smooth PDF whereas in CI approaches it represents the full state of knowledge.

The CIM approach can also be used to model dependent distributions; for two dependent variables for example the modeller may effectively specify the joint distribution explicitly (as a three-dimensional histogram), provided that the number of classes in the histograms is reasonably small. Simple graphical methods can be employed to estimate the percentage "degree of dependence" (similar to correlation) which the specified joint input distribution displays, as a check on the consistency of the input information.

B.3.1.6 Factors of Safety

The factor of safety is a factor of ignorance
[Ullman 1992, p. 232]

The factor of safety is a widely used and simple measure of the amount of contingency or "leeway" built in to a designed mechanical system in order to mitigate against the effects of uncertainties in the material properties, the geometry, the load and also uncertainty introduced by the designer's analysis methods. Many companies use factors of safety as standards. The methods used to establish the required factor of safety for a design are very approximate and rely heavily on assumptions and rules of thumb.

The safety factor, FS, is defined as:

$$FS = \frac{S_{al}}{\sigma_{ap}}$$

S_{al} = allowable strength

σ_{ap} = applied stress

If the strength and stress were both known precisely then a design for a mechanical part would simply require a factor of safety of 1 to ensure that it never failed - a larger value would be unnecessary. However, in reality these values cannot be known precisely and thus they are modelled as random variables. The factor of safety is therefore redefined as:

$$FS = \frac{\overline{S}_{al}}{\overline{\sigma}_{ap}}$$

$\overline{S}_{al}$ = mean allowable strength

$\overline{\sigma}_{ap}$ = mean applied stress

And thus, also, one cannot require that the design "never" fail - one must instead specify a desired reliability (defined as 1 - probability of failure). There are two basic techniques which are commonly used to determine whether a designed system meets the required factor of safety:

1. Evaluate the desired safety factor using rules-of-thumb and use a point-value model to arrive at S_{al} and σ_{ap}
2. Use a simplified statistical model for S_{al} and σ_{ap} based on the normal distribution.

Each is outlined below.

Safety Factors: Rules-Of-Thumb

Point values are calculated for S_{al} and σ_{ap}. Then the required factor of safety is evaluated by considering the degree of uncertainty present in the material properties, geometry and load, the uncertainty introduced by the analysis methods used and the desired reliability. A numeric value, greater than or equal to1, is assigned to each of these sources of uncertainty and the required safety factor is calculated thus:

$$FS = FS_{material} \times FS_{stress} \times FS_{geometry} \times FS_{failure\ analysis} \times FS_{reliability}$$

Simple, qualitative, rules of thumb are used to assign values to the separate factors. For example

> "if the load is not well known and/or the stress analysis method is of doubtful accuracy then FS_{stress} should lie in the range [1.4, 1.7]"

> "if the reliability is average, 92%-98%, $FS_{reliability}$ should lie in the range [1.4, 1.6]"

In [Fajdiga 1996], this "rules-of-thumb" technique (acknowledged to be the usual method used) is described as follows: The safety factor for the designed system is evaluated using a point value of S_{al} with a specified cumulative probability (often 90%) and a point value of σ_{ap} which is considered unlikely to be exceeded in operation. The required safety factor is then given by a relevant standard. Fajdiga and co-authors point out that this technique, whilst requiring very little information, does not enable calculation of the probability of failure for the designed system.

A slightly more rigorous analysis can be pursued using a simplified statistical model.

Safety Factors: Statistical Model

S_{al} and σ_{ap} are modelled as normally distributed random variables and we define FS thus:

$$FS = \frac{\overline{S}_{al}}{\overline{\sigma}_{ap}}$$

$\overline{S}_{al}$ = mean allowable strength

$\overline{\sigma}_{ap}$ = mean applied stress

Then it is possible, using probability theory and by assuming the random variables to be normally distributed, to derive an expression for FS as a function of

- allowable strength (S_{al}) statistical ratio
- applied-stress (σ_{ap}) statistical ratio
- reliability required

where the *statistical ratio* is defined as the mean divided by the standard deviation. The statistical ratio is thus a measure only of the degree of uncertainty in the allowable strength and applied stress, not of their absolute values. This is illustrated in Figure B-2 below. It can be seen that in the shaded area the system will fail (since stress exceeds strength) and thus the shaded area is equal to (1 - reliability). It can be seen that, if we assume

that both distributions are normal, then we can calculate the ratio of the means provided we know the statistical ratios and the reliability.

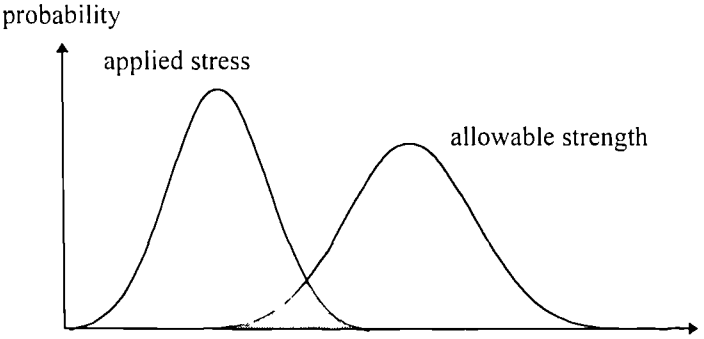


Figure B-2: Statistical model for safety factors

The statistical ratio for S_{al} can be estimated from the mean and standard deviation for the material properties - once again, generally assumed to be normally distributed.

The statistical ratio for σ_{ap} is estimated using:

- the statistical ratio for the geometry
- the statistical ratio for the load
- the level of accuracy of the method used to find the stress
- the level of accuracy of the failure analysis method used

These estimates are very approximate. The statistical ratio for the geometry is given by the tolerances. The statistical ratio for the load can be estimated from the designers estimates for minimum, most-likely and maximum load by approximating the distribution of the load to a beta distribution. These minimum, most likely and maximum values may be arrived at using rules of thumb. Qualitative rules-of-thumb are also used to adjust the statistical ratio for σ_{ap} depending upon the analysis methods used.

One obvious drawback to the method described above is that the underlying distributions are in fact often asymmetrical. The fact that load is described using a beta distribution emphasises this point. Thus the normal approximation is not really applicable.

Safety Factors: An Alternative Probabilistic Model?

In [Fajdiga 1996], the authors investigate alternatives to using safety factors. The methods they consider have the advantage over the safety factor approach that they aim to provide a value for the probability of failure - i.e. a value for the reliability of the designed system. Two methods are presented, one wholly based on analysis and simulation and one partly analytical but also making use of experimental data. The problem explored is the determination of the stress in critical sections of a vehicle chassis under uncertain operating conditions (consideration of strength is not addressed).

A set of operating conditions give rise to stresses in the designed chassis. The operating conditions are a set of time-varying random variables such as the vertical irregularities in the vehicle's path, the tyre pressure, the angular velocity of the engine, the torque of the engine, etc.

In both the analytical and the experimental method, load frequency plots (what the authors term "load collectives") are used. The load frequency plot shows the time-variant parameter transformed into the frequency domain - where frequency is measured in cycles per unit distance travelled by the vehicle. The cycle count is obtained using a counting method (e.g. the Rainflow method) [Rice 1988]. The counting method yields the number of peaks (within time T, where T is the total sample time) for each of a set of possible signal amplitudes. The number of peaks within time T can then clearly be expressed as the number of peaks per unit distance travelled, by using by the vehicle velocity.

In the analysis and simulation-based method, the problem is sub-divided into two stages. Firstly the angular velocity and the torque of the vehicle's wheels are determined from the operating conditions; thus the first stage involves modelling the vehicle transmission using transfer functions, matrices etc. Secondly, the stress in a critical section of the chassis is determined from the external loads (including the wheel angular velocity and torque); thus the second stage involves modelling the vehicle chassis using FEM or BEM. The method

relies on previous knowledge, perhaps obtained by measurement, of the statistical parameters of the random input variables (e.g. operating conditions).

Considering the first stage for example (the two stages are analogous), each operating condition is simulated at each of three levels: "heavy", "medium" or "light", where a heavy operating condition gives rise to large values for the stress and a light operating condition gives rise to small values for the stress. The definition chosen is that there is a 90% probability that the operating conditions will lighter (i.e. better) than "heavy", a 50% probability that they will be better than "medium" and a 10% chance that they will be better than "light". It is not explained in the paper how correlations between different operating conditions are handled. Previous measurements (or simulations) provide a load frequency plot for each operating condition at each level. Three simulations can then be performed - one with all operating conditions light, one medium and one heavy - yielding three time-based samples of wheel torque and angular velocity.

Similarly, three time-based samples for the stress in a critical section of the chassis could be generated by simulation. These could then be converted to give three load frequency plots, one with a cumulative probability of 10%, one 50% and one 90%. The PDF of the stress (which is required for the reliability calculation, along with the PDF of the strength) can then be estimated by a three-parameter Weibull distribution. However, Fajdiga and co-workers found that the disparity between simulated and measured stresses was too great to justify the use of this method. Thus, they propose an experimental method.

In the experimental method, the stress (in N/mm^2) was measured and digitised at a critical position on the vehicle chassis, to yield a time-variant sample. This was repeated 400 times, each time under either "light", "medium" or "heavy" operating conditions - the ratio being determined by the customer requirements. This set of 400 stress-time-domain samples was then converted to a set of 400 "load collectives" or stress-frequency plots.

A discrete set of stress frequencies F1, F2,...FK was chosen for consideration. At each stress frequency, Fi, the range of stresses which occurred was divided into a reasonable number of stress intervals and a histogram calculated. Thus a set of K frequency distributions were obtained for the stress, one for each discrete stress frequency considered. Each stress frequency distribution was then fitted to a three-parameter Weibull PDF. The average value of the three Weibull parameters obtained (β, γ and η) over all the stress frequencies was then used to define an overall Weibull PDF for stress. The validity of taking an average under these circumstances is not proven in this paper.

The reliability is then calculated by using a normal distribution for the load-carrying capability and performing a numerical integration to determine the area of overlap between the two distributions (the grey area in Figure B-2 above).

The authors conclude that analytical and simulation based techniques cannot currently be used to predict reliability with sufficient accuracy: the only sufficiently accurate technique is the testing of prototypes.

B.3.1.7 Level 1, 2 and 3 Reliability Methods

This class of probabilistic methods is usually concerned with predicting the probability of failure of a system given the probability distributions of the uncertain system parameters $\mathbf{X} = (X_1,...,X_n)$ and a function G where $G(\mathbf{X}) <= 0$ indicates the failure of the system.

More generally, there may be many G-functions each associated with a particular component. The components are then arranged into a reliability network describing the system. If two components are parallel in the network, then they must both fail in order that the path through them should fail. If they are in series, then if either component fails, the path fails. For simplicity, in what follows, it is assumed that the system is composed of a single component.

The result (the probability of failure) is usually expressed as a reliability index β, where β is defined to be the argument of the standardised normal distribution which yields (1 - the probability of failure)

$$\beta = \phi^{-1}(1 - Pr(fail))$$

i.e.

$$Pr(G(\mathbf{x}) <= 0) = 1 - \frac{1}{\sqrt{2\pi}} e^{\frac{-\beta^2}{2}}$$

The reasons for expressing the result as a reliability index rather than just as Pr(fail) are partly historical and partly practical in that the reliability index varies less sharply and more linearly with the parameters of the distributions of the X_i. This means that approximating the sensitivity of β to a parameter θ by a linear function gives reasonable results over a larger range of values of θ, than approximating the sensitivity of Pr(fail) in the same way.

The equation $G(\mathbf{X}) = 0$ defines an n-1 dimensional surface in n-dimensional space, known as the failure surface, which represents the boundary of the "failure region" and the "safe region".

Level 3 methods evaluate the exact probability associated with all points in the failure region, thus producing an exact probability of failure. The exact probability of failure, Pr(fail) – Pr($G(\mathbf{x}) <= 0$), is evaluated using the definition:

$$\Pr(\text{fail}) = \int_{G(\mathbf{x}) <= 0} \cdots \int f_{\mathbf{X}}(\mathbf{x})\, dx_1 \ldots dx_n$$

where $f_{\mathbf{X}}(\mathbf{x})$ is the joint probability distribution of $\mathbf{X}$, if the X_i are dependent.

Or, if the X_i are independent:

$$\Pr(\text{fail}) = \int_{G(\mathbf{x}) <= 0} \cdots \int f_{X1}(x_1) \ldots f_{Xn}(x_n)\, dx_1 \ldots dx_n$$

where f_{Xi} is the PDF of X_i.

Such methods require a full description of the joint probability distribution of the system parameters and are generally difficult to compute.

Level 2 methods determine a single particular point on the failure surface, known as the design point. If the system parameters are normally distributed and uncorrelated, the probability of failure can then be calculated from this single point. First order reliability methods (FORMs) approximate the failure surface at the design point to a linear surface and second order methods (SORMs) approximate it to a quadratic function (see [Der Kiureghian 1987]). The sensitivity of the reliability index to each of the system parameters is also produced as a side-product of the first order methods.

First, if necessary, the system variables $X_1,\ldots X_n$ can be mapped onto a set of independent, standardised normally distributed variables $U_1,\ldots,U_n$, spanning what is known as *u* space, using the Rosenblatt transformation as shown in [Hohenbichler 1981].

The design point is then determined. The design point is defined to be the point on the failure surface which is closest to the origin in *u* space. It can be shown that since the *u* space variables are independent and normally distributed, the reliability index as defined above is equal to the distance from the origin to the design point. Thus, having obtained the design point the reliability index is also known.

The design point is determined by approximating the failure surface to a plane and minimising the distance to the origin at the design point. This yields a set of equations which may then be solved using an iterative technique.

Level 1 methods use some pre-defined percentile values of the system parameters, rather than the full probability distributions. Thus Level 1 methods provide a "check" rather than an analysis of the system reliability. Level 1 methods may be regarded as a discretisation of Level 2 methods which are in turn an approximation to Level 3 methods.

The software package Proban® which provides both first and second order reliability methods (see [Proban 1989]) was developed at Veritas Research in co-operation with the Technical University of Munich. Development began in the seventies and continues until the present day. Proban is used in the petroleum industry, engineering consultant and design firms and the aerospace industry.

B.3.1.8 Tolerance Analysis Methods

[Chase 1991] contains a review of tolerancing methods. The area may be broadly divided into three areas : worst-case methods, statistical methods and simulation. Simulation and worst-case methods are covered elsewhere in this Appendix. There are two basic problems addressed: in statistical tolerance analysis the

distribution of the dimensions of an assembly are evaluated from the distributions of the dimensions of its components. Statistical tolerance synthesis is more computationally expensive, and involves calculating the necessary component tolerances (i.e. the parameters of the distributions for component dimensions) to achieve the desired assembly tolerance. Statistical tolerance synthesis is usually approached using optimisation techniques such as simulated annealing.

All tolerancing techniques may be divided into those which are linear and those which are not. The linear techniques assume that the sensitivity (df/dx_i below) is constant over the tolerance limits. The worst-case model is linear, simulation can model the non-linear case and statistical methods are divided. Using the linear "root-sum-squared" (or RSS) model, the tolerances of n individual components may be aggregated using the equation:

$$dU = \left[\sum_i \left(\left(df / dx_i \right)^2 T_i^2 \right) \right]^{1/2} \qquad \text{for } i = 1,...,n$$

where dU represents the variation of the dimension U of the assembly, the T_i represent the component tolerances and f(x1,..., xn) is the assembly function relating U to the component dimensions x_i. Thus df/dx_i is the sensitivity of U to the tolerance of component i. So for a one dimensional assembly, $df / dx_i = +/-1$.

The RSS model assumes that the component variations are normally distributed. Generally, the distributions are in fact flatter and skewed and RSS tends to give optimistic results when predicting the number of rejects. To account for this shortcoming a more general form is often used:

$$dU = C_f \, Z \left[\sum_i \left(\left(df / dx_i \right)^2 \left(T_i / Z_i \right)^2 \right) \right]^{1/2} \qquad \text{for } i = 1,...,n$$

where C_f is a general correction factor (typically 1.4 to 1.8), Z is the number of standard deviations required to define the variation in U and Z_i is the number of standard deviations assumed equal to the tolerance of the ith component.

The simple RSS model does not represent the case where there has been systematic tool-wear for example, and so the mean of the component distribution is no longer in the centre of the tolerance interval. The **estimated mean shift model** modifies the simple RSS with an estimated mean shift factor, m_i for each component, which represents the expected shift in mean value as a proportion of the tolerance. The RSS and worst-case estimates are combined in a proportion which depends upon the m_i, such that the result always lies between the worst-case and the RSS. If $m_i = 1$, we obtain the worst-case and if $m_i = 0$ we obtain the RSS.

The **Motorola six sigma** model is so called because it initially postulates +/- 6σ quality - meaning that the standard deviation of the component dimension distributions must be less than the required tolerance interval divided by twelve. (For a normal distribution this would mean that 99.9999998 % of component dimensions would be within tolerance i.e. there would be only .002 defects per million). However, the model includes an allowance for accumulated mean shifts. The advantage of the Motorola six sigma model is that it models both the short-term variation (with mean-shift set equal to zero) and long term variation (including mean shift). Once the target mean shift of 0.25 times the tolerance is included in the model, the quality level is reduced to +/- 4.5 σ (or 3.4 defects per million for a normal distribution).

The reader is referred to [Chase 1991] for further references on the techniques mentioned above. In [Varghese 1996] a new approach to statistical tolerance is proposed. The approach is novel in two respects. Fisrtly, the distribution used to model the input component tolerances is the finite range probability density function (FRPDF) which is a two-parameter single modal distribution which can represent skewness and mean-shift and which also has truncated tails and is thus considered particularly suitable for modelling process data. Secondly, the algorithm used to perform the analysis is numerical convolution (see Section 5). The authors demonstrate that their new algorithm is more accurate than moment-based techniques such as those described above, and faster than simulation; it is proposed that it may be sufficently fast to be applicable for tolerance synthesis.

B.3.1.9 Extension Principle

The extension principle, introduced by Zadeh in [Zadeh 1975], defines the membership function of a fuzzy set which is derived from other fuzzy sets (with known membership functions) through a known function. Let $X_1,...,X_n$ be fuzzy sets over $U_1,...,U_n$ with membership functions $\mu_1,...,\mu_n$. Let f be a function

$$f: U_1 \times U_2 \times ... \times U_n \rightarrow U$$

The extension principle tells us that we can induce a fuzzy set Y over U from $X_1,...,X_n$ through the function f and that the membership function of Y is

$$\mu_Y(y) = \sup_{x_1,...,x_n} \left\{ \min\left(\mu_1(x_1),...,\mu_n(x_n)\right) \quad f(x_1,...,x_n) = y \right\}$$
$$= 0 \qquad \text{if } f^{-1}(y) \text{ is undefined}$$

Using the definition given above, it is a non-trivial problem to calculate the membership function of Y given f and the membership functions of $X_1,...,X_n$. One approach would be to discretise the membership functions (i.e. just consider a discrete set of values for the x_i) and approximate the supremum to the maximum obtained at the discrete values. It is shown in [Dong 1987b] that this approach can yield highly anomalous results unless very large numbers of discrete values are used.

A second approach is to formulate the problem of calculating the membership function of Y as a non-linear programming problem:

Maximise $\mu_Y(y)$ subject to the constraints that

$$\mu_Y(y) \le \mu_{X_i}(x_i) \text{ for } i = 1,...,n$$
$$\text{and } f(x_1,...,x_n) = y$$

Baas and Kwakernaak (see reference in [Dong 1987b]) showed that one of the necessary conditions for a point $(x_1,...,x_n)$ to give a maximum $\mu_Y(f(x_1,...,x_n))$, is that

$$\mu_1(x_1) = \mu_2(x_2) = ... = \mu_n(x_n)$$

This suggests a method to arrive at the membership function of Y based on interval analysis, which is proposed in [Dong 1987b] and is known as the fuzzy weighted average (FWA) technique. Despite its name, the method is not restricted to calculating weighted averages of fuzzy numbers but may be used for any function f. The method uses the concept of an alpha-cut. An alpha-cut is a horizontal slice taken through a fuzzy set membership function - it is so called because it defines a crisp set called an α-level set:

$$X\alpha = \{x \quad \mu_X(x) \ge \alpha\}$$

If an alpha-cut is taken through each of the fuzzy numbers $X_1,...,X_n$ (all at the same value of alpha) we obtain n α-level sets which are intervals, $[lo_i, hi_i]$, one for each X_i (we obtain a single interval for each X_i because the X_i are fuzzy *numbers* and hence convex). The ends of the interval are defined to be the points where the membership function meets the alpha-cut. Then Baas and Kwakernaak's result tells us that the y-values with membership α on the membership function of Y are amongst the values arrived at by evaluating f for each combination of the interval end-values, e.g. $f(lo_1, lo_2, ...,lo_n)$, $f(hi_1, lo_2,...,lo_n)$ and so on. Let these y values be y_j for $j = 1,.., 2^n$. It is then clear from the definition of the extension principal given at the beginning that the y-values with membership α are $MAX_j(y_j)$ and $MIN_j(y_j)$.

Another way to describe this method, which makes its relationship to interval analysis clearer, is to consider the membership functions to be a nested series of intervals. (The intervals will be nested provided the X_i are convex). The α level of an interval is known as the level of presumption (see [Kaufmann 1985]), reflecting the fact that a wide, low-α interval estimate of a value suggests less presumption on the part of the estimator than a narrow, high-α estimate. At a particular level of presumption, the Y interval is calculated from the X_i intervals using the vertex method (described in Section 1 "Interval Analysis"). By repeating at different levels of presumption, the membership function of Y is gradually built up.

If the fuzzy sets are not fuzzy numbers and thus are not necessarily convex, there may be more than one interval for a given α-cut of a given X_i. However, the method proposed by Dong and Wong and outlined above can easily be generalised to this case.

In [Wood 1989], Wood and Antonsson suggest modelling imprecise parameter values in preliminary design as fuzzy numbers and propagating the imprecision through functions using the extension principle. They interpret the membership function at a value x as the *level of designer preference* for value x. The preference of the designer may be based upon quantifiable utility (such as minimum cost for example) as well as more subjective judgements.

The preference function representations chosen are a triangular membership function (representing an interval of possible values with a single preferred value) and a trapezoid (representing an interval of possible values containing a sub-interval of preferred values). Wood and Antonsson consider these simple representations to be adequate to represent early design preference.

Wood and Antonsson's model is intended to help designers in choosing input parameter values which satisfy a requirement on the output parameter. From the membership function of the output parameter (produced by the model) one can not only see whether the preferred values of the input parameters yield an acceptable output parameter value, but one can also immediately see how much one must alter the input parameters from their preferred values to achieve the desired output parameter value. For example, suppose there are two input parameters A and B and an output parameter Q. Suppose the required value for Q is 12, but the model shows that the value of Q with membership of 1 is 8. Thus if the designer uses the preferred values for all the input parameters then the output will not be the required value of 12. But further, from looking at the membership function for Q, the designer can also see that the preference value for the required value of 12 is, say, 0.7. It then follows, from the nature of the extension principle, that if A and B are given values with a preference of 0.7, then Q will achieve its required value of 12. Wood and Antonsson term this property of the fuzzy calculus the *backward path* - the property that the preference values of the input parameters can be deduced from the preference value of the output parameter.

Wood and Antonsson propose a method for performing uncertainty analysis (ranking the input parameters according to their contribution to the fuzziness or uncertainty of the output), based on a measure of the "fuzziness" of a fuzzy number. They call this measure the γ-level measure and it is defined so as to give a large value for a wide, bulbous, flat-topped membership function and a small value for a narrow, sharp-peaked membership function. The method comprises:

- set all of the input parameters except one to their nominal crisp values

- calculate the output parameter using the extension principle

- calculate the γ-level measure of the output parameter

Repeat the above steps for each input parameter and rank the input parameters according to the γ-level measures so obtained.

Those parameters which have a low ranking can then be safely set to their preferred values (i.e. given crisp values) without having much effect on the result.

In a later paper [Wood 1990], a direct comparison is made between the fuzzy calculus and probability as a means of representing imprecision in engineering design. In [Otto 1994] the method of imprecision is extended to include probabilistic stochastic variations which are not under the control of the designer (such as manufacturing tolerances), and also to include "possibility parameters" which may take any value within a specified range (e.g. tuning adjustments) and "necessary parameters" where the design must necessarily satisfy all values within a range (e.g. the design must deliver certain level of power over a range of motor speeds).

The extension principle, usually implemented with the FWA technique described above, has been used in fatigue analysis, particularly of civil engineering structures. In [Blockley 1979], Blockley describes a fuzzy set approach to predicting the behaviour of structures which are very poorly understood. He combines probabilistic and fuzzy representations by considering the load to be a random variable and the relationship between time-to-failure and load to be a fuzzy relation. Thus for each value of time_to_failure, the corresponding load is not a crisp number but a fuzzy number. In the first example he gives, the relationship between time_to_failure and load is derived from two relations, each modelled as fuzzy and combined using the extension principle. Having thus arrived at a fuzzy number for the time_to_failure (Y) at each load value (X) and knowing the probability of each load value, he then derives a probability distribution for the time_to_failure using Zadeh's result

$$P\{Y = y\} = \sum_{x \in X} \frac{\mu_R(y / x) \cdot P(X = x)}{P(Z)}$$

where R is a fuzzy relation between fuzzy sets X and Y (P(Z) is a normalising factor).

Another example of the application of the extension principle to engineering practice is presented in [Tee 1991]. The aim is to assess the condition of a bridge. Each of several bridge elements are given a rating from 0 to 9 by a bridge inspector, representing their condition. These ratings are represented as fuzzy rather than crisp numbers. Then each bridge element is assigned an importance factor, representing the degree of significance of that particular element to the overall condition of the bridge. Again, the weightings are fuzzy rather than crisp. The weighted average of the ratings is then calculated using the FWA algorithm described above, to give a fuzzy representation of the overall condition of the bridge. This is a typical application of FWA to engineering decision support.

B.3.1.10 Dempster-Shafer Evidence Theory

In Chapter 5 Section 5.1.4 "Mass Assignments and Pairs of Measures" we described the concepts of a mass assignment over the power set P(U) of a frame of discernment U

$$m: P(U) \rightarrow [0,1] \text{ such that } \quad \sum_{A \in P(U)} m(A) = 1 \quad \text{and } m(\varnothing) = 0$$

(known as a basic probability assignment or BPA in Dempster-Shafer theory) and a belief function
Bel: $P(U) \rightarrow [0,1]$

$$Bel(A) = \sum_{B \subseteq A} m(B)$$

The *plausibility function* is given by

$$P^*(A) = \sum_{A \cap B = \varnothing} m(B)$$

Suppose we have two pieces of evidence where a piece of evidence consists of I subsets A_i of U and a measure $M_{1,i}$ of the belief committed to each A_i. So we have:

$$m_1(A_i) = M_{1,i} \qquad i = 1,...,I$$
$$m_2(B_j) = M_{2,j} \qquad j = 1,...,J$$

where m_1 and m_2 are the BPAs corresponding to the two pieces of evidence. Dempster-Shafer evidence theory provides a means to propagate the uncertainty in these two pieces of evidence by combining them to produce a new BPA. The theory can be used to combine more than two pieces of evidence by combining them two at a time.

The sum of a BPA over the power set must equal 1, so we assign:

$$m_1(U) = 1 - \sum_{i = 1,...,I} M_{1,i}$$

and

$$m_2(U) = 1 - \sum_{j = 1,...,J} M_{2,j}$$

Thus we assign our remaining, uncommitted, belief to the entire frame of discernment - the amount of belief assigned to U represents the degree of ignorance. The combination of m_1 and m_2, denoted by $m_1 \oplus m_2$, is then given by:

$$m_1 \oplus m_2(A \cap B) = m_1(A) * m_2(B)$$

for all

$$A \in \{U, A_1, ..., A_I\} \text{ and } B \in \{U, B_1, ..., B_J\}$$

If the same subset $A \cap B$ occurs for different pairs A and B, the contribution to $m_1 \oplus m_2(A \cap B)$ from each pair (A, B) is summed.

As stated so far, the rule could result in:

$$m_1 \oplus m_2(\varnothing) = X$$

if $A \cap B = \varnothing$ for one or more pairs of sets (A, B). If $X \neq 0$, then this violates one of the definitions of a PBA, that $m(\varnothing) = 0$. Thus, $m_1 \oplus m_2(\varnothing)$ is assigned a value of 0, and all of the other values are normalised (i.e. divided by (1 - X)) to maintain:

$$\sum_{A \in P(U)} m_1 \oplus m_2(A) = 1$$

To illustrate Dempster-Shafer evidence theory, consider an example. Suppose there are four hypotheses concerning the reason that a numerically controlled machine tool has started cutting badly misshapen parts:

Back: Backlash in the axis movement
Lost: Lost motion along the axis
Pitch: Pitching movement of the spindle
Yaw: Yawing movement of the spindle

So the frame of discernment is the set U = {Back, Lost, Pitch, Yaw}. Suppose there are two pieces of evidence, the first showing that {Back, Yaw} is likely to be the problem with 60% likelihood and the second showing a 70% likelihood that {Back, Pitch, Lost} is causing the problem. Combining these two pieces of evidence using the Dempster-Shafer rule given above we obtain:

		m_2	
	$m_1 \oplus m_2$	{Back, Yaw} 0.6	U 0.4
m_1	{Back, Pitch, Lost} 0.7	{Back} 0.42	{Back, Pitch, Lost} 0.28
	U 0.3	{Back, Yaw} 0.18	U 0.12

There are two general criticisms of this theory (see [Bonissone 1987]). The first is that the computational complexity when calculating Bel(A) and P*(A) is exponential with the order of U. Although methods of reducing the complexity have been proposed, they all place constraints upon the types of hypotheses which may be represented. The second criticism is that the normalisation process, by discarding the conflicting parts of the evidence and normalising the rest, discards information. This can lead to counter-intuitive results when combining highly conflictive evidence (i.e. where X as defined above is large).

B.3.1.11 Support Logic

In Chapter 5 Section 5.1.4 "Mass Assignments and Pairs of Measures" we described the concepts of a mass assignment over the power set P(U) of a frame of discernment U

$$m: P(U) \rightarrow [0,1] \text{ such that } \sum_{A \in P(U)} m(A) = 1 \text{ and } m(\varnothing) = 0$$

and a necessary support measure $Sn: P(U) \rightarrow [0,1]$

$$Sn(A) = \sum_{B \subseteq A} m(B)$$

The possible support Sp, can then be defined by

$$Sp(A) = 1 - Sn(\overline{A})$$

The pair [Sn(A), Sp(A)], known as a support pair, may then be interpreted as an interval containing Pr(A), with the width of the interval representing the degree of certainty concerning the probability of A. Sn(A) is the necessary (minimum) support for A, given the evidence and Sp(A) is the possible (maximum) support for A given the evidence.

Support logic ([Baldwin 1987]) combines the concepts of logic programming, fuzzy sets and evidence theory. In the programming language FRIL ([Baldwin 1988]) support logic is implemented as an extension to ordinary logic programming, as exemplified by PROLOG. A database of (uncertain) facts and (uncertain) rules is stored from which (uncertain) theorems may be "proven".

Support Logic permits representation and propagation of several types of uncertainty:

incompleteness
(missing facts are assumed unknown)

imprecise definition
(facts and rules may contain fuzzy propositions)

improbability and
uncertainty as to the degree of improbability
(facts, rules and thus theorems have support pairs associated with them)

A support logic **fact** consists of a proposition and a support pair:

((<predicate> <term_1> ... <term_n>)) : (nec, pos)

The predicate may be a predicate term (as below) or a relation. Each term may be a constant (as below), a variable, a number or a list.

(has_five_doors car_under_design) : (0.8, 1.0)

An interpretation of the above fact would be

the support for the car currently under design having five doors is at least 80% and may be as great as 100%

Facts may be combined using the standard operators, conjunction (AND), disjunction (OR) and negation (NOT) to form compound propositions.

A support logic **rule** consists of a head and a body with two support pairs:

(<head> IF <body>) : (nec, pos) (nec', pos')

The head consists of a proposition and the body consists of one or more propositions combined using the standard operators. The first support pair represents the support for the head of the rule if the body of the rule is true and the second pair represents the support for the head if the body is false:

((is_a_family_car X) IF ((has_a_large_boot X) AND (has_five_doors X)))

(0.2 1.0) (0 0.1)

An interpretation for the above rule would be:

If a car has a large boot and five doors then the support for it's being a family car may be as low as 20% but it is possibly as much as 100%. So having these features doesn't provide much support for it's being a family car. On the other hand, if it doesn't have both of these features then we can be fairly sure that it isn't a family car - the support for it's being a family car lies between 0% and 10%.

The support logic rules used to evaluate support for compound statements are shown below:

NEGATION

f1 : (n1 p1)

NOT f1 : (n p) where n = 1 - p1 and p = 1 - n1

The rule for negation follows from the definition $Sp(A) = 1 - Sn(\overline{A})$ given above.

CONJUNCTION
f1 : (n1 p1)
f2 : (n2 p2)

---

f1 AND f2 : (n p) where n = n1 * n2 and p = p1 * p2

DISJUNCTION
f1 : (n1 p1)
f2 : (n2 p2)

---

f1 OR f2 : (n p) where n = n1 + n2 - n1 * n2 and p = p1 + p2 - p1 * p2

The rules for conjunction and disjunction follow from the probabilistic view of the support pair.

The language FRIL can also be used to represent fuzzy sets and hence fuzzy propositions. For example, we might define the fuzzy set *large_boot* as follows

large_boot([30:0, 40:1])

meaning that the membership function, μ_{LB}, for the fuzzy set *large_boot* takes a value of 0 at 30 cu/ft, a value of 1 at 40 cu/ft and increases linearly in between 30 and 40 cu/ft. It takes value of 0 below 30 cu/ft and a value 1 above 40 cu/ft.

We can then represent the fuzzy proposition

(has_a large_boot X)

Suppose we also have a fuzzy definition μ_{SB} of the size of the boot for the car currently under design:

size_boot([20:0, 32:1, 43:0])

(has_a size_boot car_under_design)

In order to combine such propositions with each other and with the other, crisp, rules and facts in the database we need to be able to calculate the support pair for the proposition LB:

(has_a large_boot car_under_design)

given the proposition SB:

(has_a size_boot car_under_design)

The mechanism which FRIL provides to do this is called semantic unification (in this example we are unifying the meanings of *large_boot* and *size_boot*):

SEMANTIC UNIFICATION

$$pos(LB \mid SB) = \max_x \{ \min[\mu_{LB}(x), \mu_{SB}(x)] \}$$

$$nec(LB \mid SB) = 1 - pos(NOT\ LB \mid SB) = 1 - \max_x \{ \min[1 - \mu_{LB}(x), \mu_{SB}(x)] \}$$

Thus the possible support is the maximum value of the intersection of the two sets. And the necessary support follows from the definition $Sp(A) = 1 - Sn(\overline{A})$ given above.

To calculate the support for a theorem given a set of facts and rules, all the possible proof-paths from the facts and rules to the theorem are found and each one is assigned a support pair. Each proof-path then constitutes a piece of evidence for the theorem. The pieces of evidence are then combined using either the intersection rule

(if the evidence is non-conflicting) or using the Dempster-Shafer rule described in Section 10 "Dempster-Shafer Evidence Theory" (if the evidence is conflicting).

The inference mechanism is shown below

INFERENCE RULE

(r | f1) : ((s1 u1) (s2 u2))
f1 : (n1 p1)

---

r : (n p)

where n = s1 * p1 + s2 * (1 - p1), if s1 =< s2
or n = s1 * n1 + s2 * (1 - n1), if s1 > s2

and p = u1 * n1 + u2 * (1 - n1), if u1 =< u2
or p = u1 * p1 + u2 * (1 - p1), if u1 > u2

The inference rule can be derived from the Theorem of Total Probability which states that

Pr(A) = Pr(A | B) * Pr(B) + Pr (A | NOT B) * Pr(NOT B)

Having used the inference rule to calculate the support for each available proof-path, we now need to combine the evidence. The evidence is considered to be conflicting if the support pairs for the various proof-paths do not overlap and is consistent if they do overlap.

INTERSECTION RULE

f : (n1 p1)
f : (n2 p2)

---

f : (n p) where n = max(n1, n2), p = min(p1, p2)

assuming n1 =< p2, n2 =< p1

For consistent evidence, the support for the combined evidence is assigned to intersection of all the proof-path intervals. This is the widest interval which remains consistent with all the proof paths.

DEMPSTERS RULE FOR CONFLICTING EVIDENCE

f : (n1 p1)
f : (n2 p2)

---

f : (n p)

where

n = (n1 + n2 - n1 * n2 - c) / (1 - c)
p = p1 * p2 / (1 - c)

and the conflict is given by, c = n1 * (1 - p2) + n2 * (1 - p1)

If the evidence is conflicting, the support for the three sets {f}, {NOT f} and {f, NOT f} must be considered. Values for Dempster-Shafer's basic probability assignment (BPA) function can be calculated:

m_i({f}) = ni
m_i({NOT f}) = 1 - pi
m_i({NOT f, f}) = pi - ni

and the Dempster-Shafer rule can then be applied to give the result quoted above.

To summarise, we have given an overview of the support logic rules as implemented in the logic programming language FRIL. Fuzzy or crisp propositions form the rules and facts in a database. Support pairs are associated with the rules and facts. Queries are answered by forming the support pairs for all possible proof paths and then combining the evidence from each proof path.

One difficulty which standard logics encounter when used to represent real-world knowledge and reasoning processes is an inability to represent what is known as non-monotonic reasoning. In standard logics, if a theorem may be proved from a set $A = \{H_1, ...,H_n\}$ of hypotheses then it may also be proved from any super-set of A. This property is known as monotonicity.

However, real-world reasoning does not usually display this property. The example commonly quoted is the theorem that "Ethel can fly" which may be deduced from the hypothesis that "Ethel is a bird". The addition of the hypothesis "Ethel is a penguin" however, renders the theorem false. In engineering design for example it is frequently the case that the addition of a further constraint upon the design will render a previously "good" design unusable. Since the evolution of a design model generally involves the evolution of additional constraints (for example from other parts of the design) as well as the addition of further detail to the model, the evolution of a design model may certainly be regarded as a non-monotonic process.

There are many non-standard logics (see [Smets 1988]), which have the property of non-monotonicity. Support logic is able to model non-monotonic reasoning in the sense that the addition of a fact to the database may reduce the support for a theorem.

An application of support logic to civil engineering is proposed in [Stone 1989]. The aim is build a knowledge based system (KBS) which can use information about the history of previous civil engineering projects to assess the risks associated with a new project. A database of rules is built up by learning from previous case-histories and the new project is then matched into the database to predict the risks associated with it.

Each case-history is described by an event sequence diagram (ESD), showing the sequence of events which led to the failure or success of the project. The ESD is a tree diagram representing head events each of which was preceded by all of its tail events. A set of event definitions which are sufficiently general that they are relevant to the full range of case-histories is used - for example "deficient safety culture". Uncertainty is incorporated into the data representation by associating a support pair with each event node (i.e. proposition) in the ESD, representing the degree of belief that the proposition was true of the project under consideration.

The information in the knowledge base is collected into layers. The ESDs derived directly from the case-histories form the bottom layer. The propositions in the next layer have been generalised (by the human builder of the KBS) into higher level concepts - for example "reinforcement starter bars omitted", "low strength concrete used" and "setting out error" may all be propositions in the bottom layer. In the next layer, the proposition "poor site supervision" will be used to encompass them all. Thus, when using the knowledge base, the new project can be matched into the database at whichever level of detail is available.

The algorithms required are a learning algorithm (to induce support logic rules from sets of ESDs) and a similarity algorithm (to measure the degree of similarity between the new project and the database of case-histories). The learning algorithm is based on an inductive method using fuzzy discrimination and connectivity analyses, previously applied in the area of medical diagnosis (see refs in [Stone 1989]). The support for two sets of propositions being similar is calculated as follows. If there are two sets of propositions $A = \{A_1, A_2,...,A_n\}$ and $B = \{B_1, B_2,...,B_n\}$ where the probability of A_i is a_i and the probability of B_i is b_i, then Stone defines a measure of their similarity to be

$$\Pr(\text{A and B similar}) = \frac{\sum_i \min(a_i, b_i)}{\sum_i \max(a_i, b_i)}$$

Thus, if each proposition has a support pair associated with it, $[na_i, pa_i]$ and $[nb_i, pb_i]$, then the support for the A and B being similar, [n ,p], is given by

$$n = MIN\left[\frac{\sum_i \min(a_i, b_i)}{\sum_i \max(a_i, b_i)}\right] = \frac{\sum_i \min(na_i, nb_i)}{\sum_i \max(pa_i, pb_i)}$$

$$p = MAX\left[\frac{\sum_i \min(a_i, b_i)}{\sum_i \max(a_i, b_i)}\right] = \frac{\sum_i \min(pa_i, pb_i)}{\sum_i \max(na_i, nb_i)}$$

B.3.1.12 Interval Probability Theory

In [Cui 1990], Cui and Blockley present an interval theory of probability. The probability intervals which are calculated are similar to those produced using Support Logic, but they are not the same and unlike Support Logic the calculus is based on probability theory. The vertex method (see B3.1.1 "Interval Analysis") is used to perform the interval analysis, and a parameter, ρ, called the degree of dependence is introduced. A probability function Prob is defined over the power-set of the universe of discourse:

$$\text{Prob: } P(U) \rightarrow [0,1]$$

satisfying the usual axioms of probability theory. Then ρ is defined by

$$\rho = \frac{\text{Prob}(A \cap B)}{\min(\text{Prob}(A), \text{Prob}(B))}$$

Thus ρ takes a value of zero if *A* and *B* are mutually exclusive and a takes a value of one if A is a subset of B or B is a subset of A, and it is clear that ρ provides a measure of the degree of dependence between membership of A and membership of B. In interval probability theory, an interval variable

$$PI(A) = [Sn(A), Sp(A)]$$

is defined as being the interval within which Prob(A) lies. And therefore

$$PI(\overline{A}) = [1 - Sp(A), 1 - Sn(A)]$$

The degree of dependence ρ is also generalised to be an interval number [ρl, ρu] as follows:

$$\text{Prob}(A \cap B) = \rho \min(\text{Prob}(A), \text{Prob}(B))$$
$$PI(A \cap B) = [\rho\, l(Sn(A) \wedge Sn(B),\ \rho\, u(Sp(A) \wedge Sp(B)]$$

where ∧ and ∨ denote min and max respectively. The probability interval for A ∪ B can then be calculated from:

$$\text{Prob}(A \cup B) = \text{Prob}(A) + \text{Prob}(B) - \text{Prob}(A \cap B)$$

using the vertex method to yield:

$$Sn(A \cup B) = Sn(A) + Sn(B) - \rho\, u(Sn(A) \wedge Sn(B))$$
$$Sp(A \cup B) = Sp(A) + Sp(B) - \rho\, l(Sp(A) \wedge Sp(B))$$

Similar expressions can be derived for the probability intervals of $\overline{A} \cup B$, $A \cup \overline{B}$ and $\overline{A} \cup \overline{B}$. Where there is evidence from two different proof paths, it is possible to use the calculus to combine the evidence and to obtain a measure of the conflict, but it is necessary to define the dependence interval between the two proof paths. Interval probability theory yields wider union intervals than Support Logic and, if the belief function and plausibility function values in Dempster-Shafer are regarded as defining a comparable interval, then this is wider still.

B.3.2 Closest Match Techniques

There are two, related, uses for closest match techniques in design. Firstly, database retrieval techniques where a match is sought between a query from the designer and the information stored in a data base. Secondly, nearest neighbour searches where a match is sought between the current design and a previously completed design - perhaps in order to provide initial estimates of some parameters of the new design, such as material costs or performance parameters. Initial estimates of new design parameters may also be required to provide a starting point for iterative parametric design optimisation techniques and tools.

The uncertainty arises from several sources. Firstly, it is difficult to say what is meant by "closest". Secondly, when matching a new design to a previous design, the new design attributes may be uncertain. Thirdly, when matching an exact query, there may be no exact matches or too many exact matches - so a "best" set of approximate matches is required. Fourthly, the query itself may not be exact.

In [Pahl 1984] page 4, Pahl and Beitz divide design problems into three categories, which are widely accepted:

original	- requires an original solution principle
adaptive	- adapt previous design to new task (may need original components)
variant	- solution principle and task remain the same, vary some design parameters

Information retrieval under uncertainty is clearly important in all three classes of problem. In original design, the nearest neighbour may be very different from the new design thus implying a large uncertainty in any attribute values for the new design which are deduced. However, it may still be possible to choose definitions of "nearest" (i.e. metrics) which are useful in the initial stages of original design.

Techniques which have been applied to the problem include probabilistic data retrieval, fuzzy data retrieval, learning what constitutes a good match from previous searches, and providing feedback after an unsuccessful search in the form of further evaluation procedures. Neural networks have also been used to recognise mechanical design features ([Peters 1992]).

B.3.2.1 Genetic Search as a Retrieval Technique

Genetic algorithms are a recent development in search techniques which solve problems in a similar way to the evolutionary process. A problem is solved by generating a "population" of possible solutions and introducing a "genetic operator" which produces new solutions (children) from the existing population (parents). If the population is then controlled by only allowing the "good" solutions to reproduce (i.e. by killing off the "bad" solutions), it is possible that the population will evolve into a good solution to the problem.

Thus a genetic algorithm is characterised by:

- a genetic representation of solutions to the problem
- a means of generating an initial population of solutions
- an evaluation function
- genetic operators to produce children
- values for the population parameters (e.g. population size)

In the simplest examples of genetic algorithms there are generally two genetic operators - reproduction and mutation. The reproduction operator produces children from pairs of parents, and the child solution inherits a mixture of the characteristics of each of the parents. The "mutation" operator introduces random changes into each generation.

In [Brown 1993] the author presents an automatic part-selection program which has been built using a genetic algorithm. The genetic algorithm finds a satisfactory part from the parts catalogue. The search is conducted to minimise some function (e.g. weight) under a set of constraints. Genetic search doesn't find the *optimum* solution, but it is much faster than optimisation methods and can produce a *satisfactory* solution.

B.3.2.2 Probabilistic Retrieval

The purpose of probabilistic retrieval methods is to improve the usefulness of document searches which return too many matches by ranking the returned documents. In [Savoy 1994], before introducing his own

learning scheme, Savoy gives an overview of the traditional probabilistic retrieval model. This overview is summarised below.

Searches are carried out by matching keywords (or "index terms") in the query to keywords stored in the database corresponding to each document. Documents are represented in the model by binary index terms

$$x_{ik}\ k{-}1,...,t$$

where a value of 1 for x_{ik} indicates the presence of index term T_k in document D_i and a value of 0 indicates its absence. The query is also represented as a set of index terms

$$x_{qk}\ k = 1,...,t$$

A retrieval status value (RSV) for document Di may be defined by

$$RSV(Di) = \sum_{k=1}^{t} x_{ik}.x_{qk}$$

Or, by restricting the summation to terms which occur in the query,

$$RSV(Di) = \sum_{k=1}^{q} x_{ik}$$

So the RSV is equal to the number of keywords in the query which match the document. This allows a ranking of the retrieved documents if too many are produced by the search. This technique may be refined by allowing the user to weight the keywords in the query (the x_{qk}) according to their importance as perceived by the user.

A further refinement is to give greater importance to "narrow" keywords in the query than to common ones (a "narrow" keyword being one which only occurs in a small number of the stored documents). In this case the binary variable x_{qk} is replaced by a weighted variable w_{qk}. The weight may be calculated thus, as suggested in [Sparck 1972],

$$RSV(Di) = \sum_{k=1}^{q} x_{ik}.w_{qk} = \sum_{k=1}^{q} x_{ik}.\log(n / df_k)$$

where n is the number of documents in the collection and df_k is the number of documents in which the keyword T_k occurs.

A more formal definition of w_{qk} may be obtained by using Bayes' theorem and assuming that the index terms occur independently in the relevant and non-relevant documents,

$$w_{qk} = \log[\ r_{qk} / (1 - r_{qk})\ ] + \log[\ (1 - s_{qk}) / s_{qk}\ ]$$

where r_{qk} is the conditional probability that the document is relevant given that its representation contains T_k and s_{qk} is the conditional probability that the document is not relevant given that its representation contains T_k. r_{qk} may be initially estimated as a small constant and s_{qk} could be estimated as df_k/n.

The relevance feedback process requires the user to mark each retrieved document as either relevant or not relevant after a search and send this relevance information back to the system. Robertson and Sparck Jones [Robertson 1976] show how to use the relevance information to estimate s_{qk} and r_{qk},

$$r_{qk} = (rel_k + 0.5) / (R_q + 1)$$

$$s_{qk} = (df_k - rel_k + 0.5) / (n - R_q + 1)$$

where rel_k is the number of relevant documents which include T_k and R_q is the number of relevant documents for the query.

Also, each keyword stored for the document (each x_{ik}) may be weighted to reflect its importance in describing the semantic content of the document. This weighting is known as the term significance weight. In [Croft 1983], the author suggests modifying the definition of the RSV to include term significance weight thus:

$$RSV(D_i) = \Sigma \Pr(x_{ik} = 1 | D_i).w_{qk}$$

with:

$$\Pr(x_{ik} = 1|D_i) = \begin{cases} K + (1 - K).ntf_{ik} & \text{if } ntf_{ik} > 0 \\ 0 & \text{otherwise} \end{cases}$$

where:

$$ntf_{ik} = tf_{ik} / \max_j (tf_{ij})$$

where tf_{ik} is the frequency of the keyword T_k in the document D_i, and K is a constant whose value depends upon the nature of the document collection. A value of 0.3 is typical.

Considerable work has been performed at Bath University on engineering component selection using databases ([Vogwell 1990]). One of the main differences between document and component databases is that component data is numeric rather than textual. Thus different approaches have been developed to deal with the problem of too many or too few search results in component databases. A component system called CASOC (Computer Aided Selector of Components) has been built ([Culley 1990] and [Vogwell 1992]) which provides feedback to the user both prior to a search and after an unsuccessful search. There has also been later work concerning fuzzy retrieval (see Section 3 "Fuzzy Database Storage/Retrieval", below).

At Bristol University, a combined database and hypertext system called Review has been produced (see [McMahon 1993]). Review was initially intended for storage and retrieval of engineering design information and can incorporate both numerical and textual information - information types which are supported include CAD and FEA data. The retrieval mechanism is based around attribute-value pairs which are associated with each entity stored in the database. The user may define their own attributes and may also associate several pairs, each pair having the same attribute, with a single entity. Searches are performed by applying successive "filters", where for each filter the user specifies a value or values for a particular attribute. Each filter can be applied to the results of the previous filter, producing the intersection of the results from all the filters. Thus with each successive filter applied, the number of entities returned is reduced. There is also an SQL-like query language. The advantage of the filtering mechanism over SQL is that the user is offered a pick-list of only the available attributes and value/s at each stage. Thus, it is not possible to specify a filter which doesn't return any entities, preventing "failed" searches. This is termed a "reduced search space algorithm".

B.3.2.3 Fuzzy database storage/retrieval

Fuzzy set concepts have been applied to both the data storage and the query/retrieval aspects of databases. A useful overview is given in [Buckles 1992]. We will look at fuzzy data models first and then consider fuzzy retrieval techniques.

Data models in relational databases have traditionally been homogeneous, meaning that the data items all have the same representation. The simplest method to incorporate fuzziness into the data representation of a relational database is to associate a membership value with each row in a relational table, thus the data model remains homogeneous. For example, suppose a database, used in the design process, contains representations of automotive components. A table might relate each component to a "trim level" , an integer from 1 to 5 representing the luxury level of the cars the component might be used on. In this example, the membership value might be used to denote the degree to which the component is specific to a particular trim level:

Part Description	Trim Level	Membership Value
walnut dashboard	5	1.0
hard-feel polypropylene dashboard	1	1.0
leather gear-stick top	3	0.5
soft-feel polypropylene dashboard	3	1.0

Thus, a walnut dashboard will definitely only be used on the most luxurious cars and the hard-feel polypropylene version will definitely only be used on the most economical models. A leather gear-stick top however, is only loosely associated with middle-of-the-range models. An example of a component which is un-associated with any trim level and so is missing from the relation altogether might be the crank-shaft.

The homogeneous fuzzy data model developed by Anvari and Rose and also that due to Buckles and Petri (see refs in [Buckles 1992]) use the idea of a similarity relation. The base representation is a set of fuzzy

numbers or linguistic variables. The values stored in the table are subsets of the base set. The extent to which a particular pair of base values are interchangeable is stored in a *similarity relation*. The example below is adapted from [Buckles 1992].

Design Option	Estimated Material Cost
A	moderate
B	{high, extreme}
C	{moderate, high}
D	{low, medium}

	Extreme	High	Moderate	Medium	Low
Extreme	1	0.81	0.81	0.75	0.25
High	0.81	1	0.90	0.75	0.25
Moderate	0.81	0.90	1	0.75	0.25
Medium	0.75	0.75	0.75	1	0.25
Low	0.25	0.25	0.25	0.25	1

Clearly the similarity relation can only be used for finite, discrete base sets. A measure of similarity which can be used for continuous and infinite base sets of fuzzy numbers is α-proximity (see refs in [Buckles 1992]):

> Fuzzy numbers A and B are α-similar if given $\beta \in [0,1]$, $x \in (A \cup B)_\alpha$ and $y \in (A \cup B)_\alpha$ and $z = \beta x + (1 - \beta)y$ then $z \in (A \cup B)_\alpha$. Where $(A \cup B)_\alpha$ denotes the alpha level set obtained from the fuzzy set union of A and B.
>
> Two fuzzy numbers A and B are α-proximate with respect to a set P if there exist fuzzy numbers X1,...,Xn ∈ P such that A is α-similar to X1 is α-similar to.... is α-similar to Xn is α-similar to B

The two representations described above support fuzzy data values but do they not support uncertainty between values. A simple approach to representing uncertainty between values is to store a possibility distribution over the base representation. This allows a heterogeneous mixture of types of uncertainty in the data value to be stored:

- not applicable (possibility = 0 everywhere)
- crisp interval
- fuzzy interval
- known

If the data itself is certain, but fuzzy queries are required, then single base representation values can be stored. A second, binary, relation specifies the degree of membership of each base representation value for each fuzzy query term, such as TALL below:

Name	Height
Jim	6-1
John	5-9
Jane	5-3

The PERSON relation

Height	Membership
5-9	0
5-10	0.6
6-0	1
7-0	1

The TALL relation

An overview of retrieval techniques for relational databases with fuzzy data is given in [Buckles 1992]. [Buckles 1991] presents an approach to the incorporation of uncertainty into object-oriented database representations based on the concept of a degree of membership in class inheritance relationships.

At Bristol University work has been carried out on fuzzy component retrieval using FRIL (an implementation of support logic), see [Lehane 1990]. Work has also been carried out at Bath University on automated selection of ill-defined components using fuzzy sets, see [Wood 1994].

B.3.3 Best Alternative Techniques

Much of design is characterised by decision making under uncertainty. In the conceptual design phase, discrete choices arise between alternative concept variants. In the embodiment and detailed design phases, choices of (often continuous) parameter values must be made. In either case, the design must be evaluated to provide a basis for a decision, and there is often a trade-off to be made between technical utility and cost.

The area of **design optimisation** is concerned with selecting a set of design parameters which optimise the goodness or utility of the design, subject to some constraints. This is often represented in the literature as "searching the parametric design space", and generally takes place during the embodiment and detailed design phases. The cost-benefit trade-off is built in to the utility function. This area can be divided into deterministic and probabilistic techniques. The **Taguchi method** is a design optimisation technique which has become very widely accepted in industrial practice.

The area of **decision support** is concerned with the evaluation of alternatives, again using a utility function, often with multiple objectives.

As mentioned in the introduction, the sources of uncertainty in "best alternative" techniques may either be uncertainty about what is "best" or may be uncertain design attributes parameters. The techniques outlined below are examples of design evaluation and optimisation methods which include examples of both types of uncertainty.

B.3.3.1 Deterministic Design Evaluation and Optimisation

In deterministic design evaluation and optimisation, the design parameters are treated as point values - the representation of the parameter values does not include any uncertainty.

The problem is to minimise or maximise a function, say $f(x_1,...,x_n)$, subject to a set of m constraints $g_j(x_1,...,x_n) \leq c_j$ for $j=1,...,m$.

The simplest case is when there are no constraints and the set of n simultaneous equations:

$$\frac{\partial f}{\partial x_i} = 0 \qquad \text{for } i = 1,\ldots,n$$

can be solved directly (for example if the $\frac{\partial f}{\partial x_i}$ are linear in the x_i). If this is not possible then an iterative technique may be used. Hill-climbing (or valley-descending) methods involve changing the estimated solution in a direction in which the value of f increases (or decreasing). In the method of steepest descent, the estimated solution is changed in the direction of the gradient of f. Newton's method, the method of conjugate gradients and quasi-Newton methods are common iterative techniques for the unconstrained problem (see [Beale 1988]).

The constrained problem where f is a linear function of the x_i and the constraints are also linear (the g_j are all linear functions of the x_i) are solved using linear programming techniques such as the simplex method. If the objective function or the constraint functions are non-linear then the problem can often be reformulated using Lagrange multipliers. Other techniques involve separating the objective function into a linear combination of non-linear functions, each of a single argument. If this is not possible then reduced gradient methods may be used (see [Beale 1988]).

Several groups have built knowledge based systems to perform parametric design optimisation using iterative design techniques. In such systems, a design-evaluate-redesign loop is repeated until the evaluation shows that the design satisfies its constraints and performance criteria. The initial values and the redesign step are guided by expert knowledge stored in the system. The evaluation step will usually involve a simulation of the design. See [Nicklaus 1987] for details of ENGINEOUS, see [Dixon 1987] for details of DOMINIC and DOMINIC I and see the references in [Ramachandran 1992] for details of DPMED.

In [Ramachandran 1992], the authors consider four alternative strategies for choosing the initial parameter values for iterative parameter design using an existing database of previous designs. The four strategies are

- Always use the same initial values (i.e. non-library method).

- Use the closest-neighbour from the existing designs. The distance metric used is a weighted sum of the normalised difference between the problem parameters.

- Fit linear functions, using the method of least squares, between the problem parameters and each design parameter, over all the designs in the database. Use the functions to predict the initial design parameters.

- Fit linear functions as above, but only fit to the n nearest neighbours, where the user specifies n.

A statistical comparison was performed of the number of iterations required to complete a design using each initial design strategy in turn. It was concluded that the number of iterations required was approximately inversely proportional to the number of designs in the library for all the library methods. Results were similar for all three library methods and since the nearest neighbour is the most computationally efficient, this would generally be the most appropriate technique. Since the library designs were randomly generated, with evenly spread problem parameters, these results are unsurprising. If the library designs were clustered around particular problem parameter values and the new design lay some distance from these clusters, then the computational price of curve fitting, possibly with non-linear curves, would be more justifiable.

Even when the design parameter representation does not include uncertainty, when the search of the design space may not be exhaustive, due to the combinatorial complexity of the design space, we have to address uncertainty that we have found the "best" solution. In [Quadrel 1993], the author describes a system called Anarchy which searches the design space using simulated annealing.

Anarchy uses the *asynchronous team* (A-team) strategy to iteratively produce building design solutions. The A-team consists of a network of autonomous, asynchronous *agents,* which are simple self-contained software tools each with their own particular goals. The A-team strategy differs from the OO paradigm because the information in the model is all global - information is broadcast and all agents receive generally broadcast messages. The mechanism for the co-operation of the agents (to prevent them from each simply each concentrating on the part of the design space which suits their own goals) is that they broadcast the results of their searches and change their search strategy depending upon the results received from other agents.

The *simulated annealing* search technique is an extension of valley-descending, to cope with the case that the solution found is a local not a global minima. In simulated annealing, perturbations are initially permitted in all directions, uphill or downhill. This corresponds to the high temperature phase of physical annealing. As the temperature is lowered, the perturbations are gradually reduced to the downhill direction only.

In the Anarchy implementation, there are three objective functions rather than one. Rather than simply taking a weighted average to calculate the "height" of the solution, a *dominance relationship* is used. If solution A is better than or equal to solution B using all the objective functions, then A is said to *dominate* B. If A neither dominates, nor is dominated by, B then A is said to *contend with* B. The set of all non-dominated solutions is known as the *Pareto set* and comprises the best set of solutions.

The simulated annealing algorithm in Anarchy has been adapted to deal with sets of solutions rather than following a single solution path. A set of accepted solutions is maintained. New solutions are generated by perturbing an existing accepted solution. A non-dominated set (ND set) of solutions which haven't been dominated yet is also maintained. If the new solution is dominated by a member of the ND set (this corresponds to being uphill from the existing lowest solution in normal simulated annealing), then it may still be accepted (for further perturbation) with a probability given by e^{-DT} where T is the annealing temperature. D, which would represent the change in height in normal simulated annealing, is the *distance* from the new solution to the ND set.

The distance function chosen is designed to give a smaller result, making the new solution more likely to be accepted, if the new solution is in a different part of the solution space from most of the ND set (i.e. a sparsely visited part of the solution space). Consider a vector, where each component corresponds to one of the objective function values. This defines a point in performance space. Let the *target point* be the point where each objective function individually takes the largest value over all the solutions in the ND set. The distance function is then defined to be (a - b) where a is the Euclidean distance from the new solution to the target point and b is the Euclidean distance from the target to the nearest member of the ND set.

B.3.3.2 Probabilistic Design Evaluation and Optimisation

A general formulation of the probabilistic design optimisation problem is presented on pp. 383-386 of [Siddall 1983]. We wish minimise or maximise a function, say:

$$f(x_1,...,x_n, a_1,...,a_q)$$

subject to a set of m equality constraints:

$$g_j(x_1,...,x_n, a_1,...,a_q, r_1,...,r_p) = 0 \text{ for } j=1,...,m$$

and L inequalities

$$h_j(x_1,...,x_n, a_1,...,a_q, r_1,...,r_p) \geq 0 \text{ for } j=1,...,L.$$

In probabilistic design optimisation and evaluation, some or all of the design parameters (x_i above) are treated as random variables. Furthermore, there may also be parameters which will not be adjusted during optimisation which are treated as random variables (a_i above) and some requirements (constraint limits, r_i above) may also be represented as random variables. During the optimisation, the design parameters are adjusted to maximise (or minimise) the expected value of the utility function subject to the expected value of the constraints being satisfied. The use of the expected value here is arbitrary - one could also use the mean or the mode or some measure which includes the spread of the resultant distribution (as in Taguchi's method). Using the expected value, we can formulate the problem as:

$$\text{maximise } E(f(x_1,...,x_n, a_1,...,a_q))$$

subject to a set of m equality constraints:

$$E(g_j(x_1,...,x_n, a_1,...,a_q, r_1,...,r_p)) = 0 \text{ for } j=1,...,m$$

and:

$$P(h_1() \geq 0 \text{ AND } ... \text{ AND } h_l() \geq 0) \geq D$$

where E(x) represents the expected value of x and P(X > x) represents the probability that X will be greater than x. D is known as the *dependability* of the design.

Having formulated the design problem in this way, in principle any of the probabilistic methods for propagating uncertainty (e.g. simulation) can then be used in conjunction with a deterministic optimisation technique. The computational complexity of such methods is clearly very high. Some examples of probabilistic optimisation in mechanical design are given in [Haugen 1980].

B.3.3.3 Taguchi's Method

Taguchi's method ([Taguchi 1986]) is a design optimisation methodology where the utility of the design is defined as its quality. The trade-off between quality and price is recognised and so it is necessary to evaluate the monetary value of quality loss - this is done using Taguchi's *loss function.*

Taguchi defines quality loss as the complete loss to society after the product is shipped. Losses such as functional failure, operating and maintenance costs and harmful effects such as pollution are all included. Thus defective goods which are not shipped and so do not reach the customer (used by some companies as a measure of "quality loss") are considered as an increased cost, not as a quality loss.

The second strand to Taguchi's philosophy is the concept of *off-line quality control.* This arises from the fact that many of the factors which cause quality loss are extremely difficult to predict or control "on-line". Thus Taguchi suggests that the nominal values of design parameters should be chosen so as to minimise sensitivity to such factors and so produce a *robust design.*

The causes of quality loss are termed *noise factors* and, for functional failure, are separated into *internal noise factors* consisting of deterioration of the product and manufacturing imperfections and what are termed *external noise factors* meaning environmental factors such as operating temperature, humidity and so on. Taguchi argues that it is generally more economical to reduce quality loss by choosing nominal parameter values which are less sensitive to noise factors rather than by reducing tolerances.

Generally the quality characteristic of a product will be related to several product parameters by a non-linear function. Thus, there will generally be several sets of parameter values which yield an acceptable quality characteristic value, and the choice between them can be made on the basis of minimising sensitivity to the noise factors.

This process of finding a set of design parameters (termed *control factors*) which will maximise the robustness of the products is known as Taguchi-class experimental design. The three central tools are the *loss function*, the *signal-to-noise ratio* and testing alternative designs using *orthogonal arrays*.

There are strong parallels between Taguchi's terminology and units and those used in dynamic signal processing - particularly his distinction between signal level (parameter value) and noise (stochastic variation in parameter value). For example, he expresses the signal-to-noise ratio in decibels. There are parallels between the process of Taguchi-class experimental design and Fourier analysis of an electrical signal (p.51 [Phadke 1989]).

The loss function is defined to be

$$L(y) = k(y - \tau)^2$$

where y is the product's functional quality characteristic, τ is its nominal (i.e. target) value and k is a proportionality constant. This form is chosen because it evaluates to zero when y = m and it also takes its minimum value (i.e. its derivative is zero) at this point.

In choosing a set of experiments, i.e. a set of values for the control factors, it is not practical to test all levels of all control factors. Suppose there are four controllable factors in a design, A, B, C and D. For each factors we wish to consider three possible values or *levels*, 1,2 and 3. To test all possible levels of all the factors, we would need to perform 34 = 81 experiments. Instead, we choose a set of 9 experiments as shown in the table below

Experiment No.	factor A	factor B	factor C	factor D	η
1	1	1	1	1	-20
2	1	2	2	2	-10
3	1	3	3	3	-30
4	2	1	2	3	-25
5	2	2	3	1	-45
6	2	3	1	2	-65
7	3	1	3	2	-45
8	3	2	1	3	-65
9	3	3	2	1	-70

This table (known as the standard orthogonal array L9) has the property that for any pair of columns, any combination of factor levels occur and they all occur an equal number of times. (In L9, they all occur exactly once). This property is known as orthogonality and it can be shown (p 277 Phadke 1989]) that it is equivalent to a definition of orthogonality based upon the inner vector product of vectors corresponding to the columns equalling zero.

For each experiment in the array, we will make n observations of the quality characteristic, y, and calculate the *signal-to-noise ratio*, η. The signal-to-noise ratio combines the mean level of y and its variation in a single metric - our objective will be to maximise S/N. It is based upon the ratio of the mean to the standard deviation. There are three types of S/N ratio - the *nominally best* (N type), the *smaller-the-better* (S type) or the *larger-the-better* (L type):

N type $$\eta = 10\,\mathrm{Log}_{10}\frac{\mu^2}{\sigma^2}$$

where $$\mu = \frac{1}{n}\sum y_i \quad \text{and} \quad \sigma^2 = \frac{1}{n-1}\sum (y_i - \mu)^2$$

S type $$\eta = -10\,\mathrm{Log}_{10}\left[\frac{1}{n}\sum y_i^2\right]$$

L type $\eta = -10\,\mathrm{Log}_{10}\left[\frac{1}{n}\sum 1/y_i^2\right]$

The quality characteristic y is always assumed to be continuous and non-negative (i.e. takes values between 0 and ∞). The ratios are arrived at by minimising the average quality loss, expressed in decibels.

Next, we calculate m, the average η over all the experiments, and the effect of a factor level is defined as the deviation it causes from this overall mean. Thus, in the above example, the effect of setting factor A to level 1 is estimated as

$$mA1 - m = (-20 - 10 - 30)/3 - 41.7 = 21.7$$

This process of estimating the effect which each factor level has on the signal-to-noise ratio is known as analysis of means (ANOM). The validity of this analysis is dependent upon the relationship between η and the factor levels being adequately approximated by the additive model

$$\eta = \mu + a_i + b_j + c_k + d_l + e$$

where a_i is the effect on the signal-to-noise ratio due to setting factor A to level i and so on, and e is the error. In other words, the effects of the factors must be separable. We constrain

$$a_1 + a_2 + a_3 = 0$$

(and similarly for b and c) as part of our definition of the a_i. Thus mA1 is an estimate of $\mu + a_1$ and so on. The error term, e, includes both the error due to the additive model not being exactly correct and the error due to the experiments not being exactly repeatable.

In [Tsai 1993], the authors propose a conjugate array experimental layout as an alternative to orthogonal arrays. This is arrived at by considering a non-additive model for the signal to noise ratio, which includes interaction terms between the controllable factors. Thus the conjugate array layout provides more accurate results in the case of significant interaction between the controllable factors. This is illustrated in the paper using a simple example of optimising the dimensions of a press-fit fly-wheel and shaft assembly with respect to the compression stress in the shaft.

Provided the additive model is valid, the optimum level for each factor can be deduced from the effects calculated in the ANOM analysis. In the example above the average values of η for each factor level are given by

Factor	1	2	3
A	**-20**	-45	-60
B	**-30**	-40	-55
C	-50	**-35**	-40
D	-45	**-40**	**-40**

implying that the optimum levels are (A, B, C, D) = (1, 1, 2, 2) or (1, 1, 2, 3).

The final stage in the Taguchi method is analysis of variance or ANOVA. This is carried out to determine the relative effect of the different factors. Drawing on the signal processing analogy, if the nine observed values of η (η1,...,η9) are like the time domain signal then the effects of the four factors are like four harmonic components of a Fourier decomposition of the signal. The total power present in the signal is analogous to the *grand total sum of squares*

$$\text{grand total sum of squares} = \Sigma\, \eta_i^2$$

The DC power is analogous to the *sum of squares due to mean*

$$\text{sum of squares due to mean} = \Sigma\, m^2 = 9m^2$$

Each harmonic component is analogous to the relative significance of each factor and is given by, for example

sum of squares due to A =

$$3\,(\,mA1 - m\,)^2 + 3\,(\,mA2 - m\,)^2 + 3\,(\,mA3 - m\,)^2 = 2450\ (dB)2$$

These values are shown in the *sum of squares* column of an ANOVA table, along with estimates of error terms.

In the discussion above, it was assumed that all values of all the noise factors were tested experimentally. This is generally not the case and the noise factors are also experimentally tested using an orthogonal array (called the outer array) - the control factor array is then known as the inner array. The N-type S/N ratio is then given by

$$\eta(\overline{P}_j) = -10\,Log_{10}\left(\frac{1}{n}\sum_{i=1,..n}(y(\overline{N}_i,\overline{P}_j)-\tau)^2\right)$$

where $\overline{P}_j$ is the control factor values for an experiment and there are n noise factor experiments $\overline{N}_i$ i=1,...,n. Once again, it can be seen that the signal-to-noise ratio is a metric which combines closeness to target value and variance.

To summarise, in Taguchi-class experimental design, optimum parameter values are defined to be those which give the least variance in the product quality characteristic. Optimum values are estimated from a small number of experiments by using the idea of orthogonal experimental arrays and assuming the additive model. The techniques have been widely applied in industry.

The Taguchi method assumes that all values of all noise factors are equally likely (an equal number of experiments are performed at each level of each factor and the results are equally weighted) and so does not model probability. Neither can joint constraints between control factors be modelled in Taguchi-class experimental design, since the method depends upon freedom to choose a set of experimental points in the design space - the experimental layout given by the orthogonal array may violate such a joint constraint.

In [Otto 1993], Otto and Antonsson suggest extensions to the Taguchi method which address these shortcomings and they also extend the model to include necessity and possibility (defined in [Wood 1989] and [Wood 1990]). A necessity constraint requires a design to meet every value within a particular range (for example the set of speed and torque values which must be met by a motor design). A possibilistic parameter is a parameter with a set of values any of which might be used - for example user adjustment variables such as seat position in automotive design.

Otto and Antonsson suggest incorporating probability into the Taguchi method by modifying the signal-to-noise ratio to include Pr_i, the probability of the noise factors taking the values $\overline{N}_i$ (the values for experiment i):

$$\eta(\overline{P}_j) = -10\,Log_{10}\left(\sum_{i=1,..n}(y(\overline{N}_i,\overline{P}_j)-\tau)^2 \times Pr_i\right)$$

They observe that if there is a necessity interval in the design, then the design should be evaluated for the worst case from within the interval. Thus the signal-to-noise ratio is further modified to become:

$$\eta(\overline{P}) = -10\,Log_{10}\left(MAX_{k=1,...,K}\left[\sum_{i=1,..n}(y_{ik}(\overline{P})-\tau)^2 \times Pr_i\right]\right)$$

if there are K levels in the necessity experiments and n levels in the noise factor experiments.

If there are possibilistic variables present in the design then there are two cases to consider. In the first case, the possibilistic uncertainty occurs before the probabilistic uncertainty and in the Taguchi method the possibilistic variable must be treated as a control factor. But if the possibilistic uncertainty occurs after the probabilistic uncertainty then the possibilistic variable is a *tuning parameter* and can be used to overcome the effect of the probabilistic variables - for example post-manufacturing adjustment variables. The signal-to-noise ratio can then be modified to become

$$\eta(\bar{P}) = -10\,\mathrm{Log}_{10}\left[\sum_{i\ 1,..n} \mathrm{MIN}_{k=1,....K}\left((y_{ik}(\bar{P}) - \tau)^2\right) \times \mathrm{Pr}_i\right]$$

Finally, Otto and Antonsson suggest a further extension (using a boundary method) to include joint constraints between control factors. The method is based upon subtracting a large number from the signal-to-noise ratio if a constraint is violated.

Otto and Antonsson also compare Taguchi's method (orthogonal arrays) with other conventional methods. They conclude that Taguchi's method requires fewer experimental points but is less accurate than simulation (e.g. Monte Carlo) as a way of approximating the effects of the noise factors. The search technique is slower than a hill-climbing technique would be but unlike such techniques, it does not require a good starting point.

B.3.3.4 Decision Support

In this section we consider design evaluation techniques where the result of the evaluation is not explicitly used to optimise the design but is simply presented to the designer as an aid to decision making.

On pages 118-139 of [Pahl 1984], the authors suggest a weighted objectives method (WOM) for evaluating concept variants. The method combines the ideas of cost-benefit analysis (originally attributed to [Kesselring 1951]) and the combined technical and economic evaluation technique attributed to [Zangemeister 1970].

A set of objectives are identified which are arranged into a hierarchy, with each objective having as its sub-nodes those objectives which contribute towards it. Each objective is assigned a weighting which reflects its importance. Parameters are then associated with each objective, for example the objective "low fuel consumption" may have an associated parameter "fuel consumption in mpg". The variants can then have magnitudes assigned to each parameter (magnitudes may be qualitative, such as "good", or they may be quantitative). These magnitudes are then normalised to give values between, say, 0 and 10 for all parameters and thus each variant can be evaluated by using the weighted objective tree to aggregate the parameter values. The cost-benefit trade-off can be illustrated by choosing technical utility and cost as root objectives, and then plotting technical rating against economic rating for each variant. The resultant diagram is known as a strength diagram (or s-diagram).

Decision support theory ([Keeney 1976]) is used to determine the relative merit of a set of possible decisions. The paradigm is simple:

Probability trees are generalised to decision trees which have nodes for decisions taken as well as nodes for uncertain events. The probability of each uncertain event is marked on the tree and thus each leaf node has both an outcome and a calculated probability associated with it. The decision-taker can then assign a utility value to each leaf-node of the tree which reflects the relative merit of the outcome as well as his attitude to risk. For example, a cautious engineer may assign a very low utility to a very low-cost design which he only has a medium probability of obtaining, whereas a more enthusiastic risk-taker would assign a higher utility to such an outcome.

In order to analyse a problem in this way, it is necessary to discretise the probability distributions of the decision variables so that the tree has a finite number of branches. If there are N decision variables, each discretised to M values, the decision tree will have M^N leaf nodes. The complexity of such models clearly soon becomes prohibitive. There are two solutions to this problem (pp. 192-198 [Morgan 1990]):

- Use sensitivity analysis to identify the most critical decision variables and only include these in the decision tree.

- Use the method of Discrete Probability Distributions (DPD - also see Section 5). The idea of this method is to condense the total number of options back to M after each node of the tree. The method assumes that the options are independent of each other. Also, having condensed the distributions after each node, it is no longer possible to tell what effect each decision had on the result (perform sensitivity analysis).

Although the decision variables are discretised, the chance variables do not need to be and so generally the decision-maker will need to choose between PDFs associated with each set of decisions. He or she may not need to formally define a utility function in order to make a choice if the outcome is defined in terms of just

one variable, such as cost. However, if the outcome involves several parameters then it would be very difficult to choose the "best" of several multi-dimensional PDFs without using a utility function.

The problem of how to construct such a utility function is addressed by the area of multi-attribute utility theory.

In [Thurston 1991a] the author argues that the WOM, described above, is flawed in two respects. Firstly, by the assumption that the objectives are linear functions of the parameter values. Secondly, the WOM does not account for the trade-offs which designers are often willing to make between parameter values - the importance weighting of parameter A may depend upon the value of parameter B.

She uses multi-attribute utility theory to define a methodology for the evaluation of design alternatives (MEDA) which allows the objectives to be non-linear functions of the parameter values and which also allows trade-offs between parameter values to be quantified. The non-linearity in the utility function may be interpreted as a measure of risk-aversion, since it implies that a loss of ΔA in the parameter value causes a larger change in the utility than a gain of the same amount.

A series of questions are asked of the decision-maker to elicit the acceptable trade-off between design parameters and the utility function for each parameter in a quantitative form. The question format is based on the *certainty equivalent method* described in [Keeney 1976]. From the answers to these questions, a multi-attribute utility function is constructed and alternative designs can thus be ranked. The sensitivity of the ordinal ranking to the design parameters can also be determined as can the trade-off between any two design parameters which will leave the utility function at a constant value.

The questions asked to determine the utility function for a particular parameter are described here. Suppose the attribute under consideration is cost, so the smaller the value the higher the utility, and the range of plausible values are £5 to £100. The question is asked:

"Which would you prefer, a certain cost of £30 or a 40% probability of £5 and a 60% probability of £100?"

By repeating the question for different probabilities, the probability, P, is determined at which the decision maker is indifferent between the two options. It could then be said that the utility of a cost of £30 is equal to

$$U(£5) * P + U(£100) * (1 - P) = P$$

since the utility of £5 is one and the utility of £100 is zero. Other points on the utility function are obtained by repeating the process with other values of the certain cost.

If the attributes are utility independent (the form of the utility function for attribute A is unaffected by the value of attribute B), then the multi-attribute utility function for the i'th design alternative is of the form:

$$U^i(X) = \frac{1}{K}\left\{\left[\prod_{j=1}^{J}(Kk_jU_j(x_j)+1)\right]-1\right\}$$

where K is a scaling constant given by:

$$1+K=\prod_{j=1}^{J}(1+Kk_j)$$

$X = (x_1,...,x_J)$ is the vector of the design attributes. The k_j and the U_j are the scaling constant and the single-attribute utility function for attribute x_j and are assessed from the responses to the questions asked of the decision-maker.

The MEDA methodology is applied to a choice between alternative designs for automotive structural frame and body-skin systems. The design alternatives which were considered were steel, steel-frame with polymer-composite skin, aluminium and hybrid. The decision-makers were taken from six automotive companies. The desirability of various attributes such as capital cost, weight, design flexibility and corrosion resistance to each company were evaluated. The overall ranking of the designs and the attributes trade-offs which were acceptable were compared for each company.

The major disadvantage of the MEDA methodology over simpler design evaluation methods such as the weighted objectives method is the complex and time-consuming information-gathering exercise required to construct the utility function.

In [Thurston 1991b], the model is extended and the attributes are modelled as independent random variables with PDFs given by the beta distribution. The beta distribution is chosen because the PDF can easily be constructed from minimum, maximum and most-probable values elicited from the estimator. The expected value for the utility of a single attribute x_j is then given by:

$$E(U_j) = \int_{x\min}^{x\max} U_j(x_j) f_j(x_j) dx_j$$

where f_j is the PDF. Thus it can be seen that a combination of a non-linear U_j (recall that this implies risk aversion) and an increase in the spread of the PDF will cause a decrease in expected utility. Comparing this approach with the Taguchi method, it can be seen that the signal and the noise are here considered to have frequencies of the same order - variations in the attribute value (due to uncertainty) are only of interest when they are sufficiently large to cause a change in the overall utility.

B.4 References

[Andrews 1993], J.D. Andrews and T.R. Moss, "Reliability and risk assessment", Harlow: Longman Scientific and Technical, 1993.

[Bahrami 1992] A. Bahrami and C. H. Dagli, "Design retrieval by fuzzy neurocomputing", JED, vol. 3, no. 4, pp. 339-356, 1992.

[Baldwin 1987] J. F. Baldwin, "Evidential support logic programming", Fuzzy Sets and Systems, no. 24, pp. 1-26, 1987.

[Baldwin 1988] J.F. Baldwin, T.P.Martin, B.W.Pilsworth, "FRIL Manual, version 4.0". FRIL Systems Ltd, Bristol ITeC, St. Annes House, St.Annes Road, Bristol, BS4 4AB

[Beale 1988] E. M. L. Beale "Introduction to optimization", Chichester: John Wiley & Sons, 1988.

[Beck 1977] James V. Beck and Kenneth J.Arnold, "Parameter estimation in engineering and science", New York: John Wiley & Sons, 1977.

[Beheshti 1993] R. Beheshti, "Design decisions and uncertainty" Design Studies, vol. 14, no. 1, pp. 85-95, Jan. 1993.

[Blockley 1979] D. I. Blockley, "The role of fuzzy sets in civil engineering" Fuzzy Sets and Systems, vol. 2, pp. 267-278, 1979.

[Bonissone 1987] P. Bonissone, "Reasoning, plausible" in Encyclopaedia of Artificial Intelligence Volume 2, ed S. C. Shapiro. New York: John Wiley & Sons, 1987, pp. 849-863.

[Brown 1993] D. R. Brown and Kuo-Y. Hwang, "Solving fixed configuration problems with genetic search", Res Eng Des, vol. 5, pp. 80-87, 1993.

[Buckles 1991] B. P. Buckles, R. George, F. E. Petry, "Towards a fuzzy object oriented data model" in Proceedings of NAFIPS Conference 1991, University of Missouri, Columbia, pp. 90-93, May 1991.

[Buckles 1992] B. P. Buckles, F. E. Petry, "Fuzzy databases" in Encyclopaedia of Artificial Intelligence Second Edition Volume 1, ed S. C. Shapiro. New York: John Wiley & Sons, 1992, pp. 508-515.

[Cartmell 1993] M. P. Cartmell, A. P. Fothergill, and J. K. W. Forster, "Theoretical foundations for a design brief expansion system" JED, vol. 4, no. 3, pp. 221-248, 1993.

[Chase 1991] K. W. Chase and A. R. Parkinson, "A survey of research in the application of tolerance analysis to the design of mechanical assemblies" Res Eng Des, vol. 3, pp. 23-37, 1991.

[Choobineh 1992] F. Choobineh and A. Behrens, "Use of intervals and possibility distributions in economic analysis" J. Opl Res. Soc., vol. 43, no. 9, pp. 907-918, 1992.

[Cooper 1987] Cooper, D. F. and Chapman, C., "Risk Analysis for Large Projects: Models, Methods and Cases", John Wiley and Sons, Chichester, ISBN 0 471 91247 6

[Cox 1961] R. Cox, "The algebra of probable inference", Baltimore: The Johns Hopkins Press, 1961.

[Croft 1983] W. B. Croft, "Experiments with representation in a document retrieval system", Information Technology: Research and Development, vol. 2, pp. 1-21, 1993.

[Cui 1990] Cui, W. and Blockley, D. I., "Interval Probability Theory for Evidential Support", International Journal of Intelligient Systems, vol. 5, pp. 183-192, 1990.

[Culley 1990] S. J. Culley and J. Vogwell, "The role of feedback in computer aided component selection systems" CSME Mechanical Engineering Forum, Toronto, 1990.

[Davenport 1970] Davenport, W. B. "Probability and Random Processes: an introduction for applied acientists and engineers", McGraw -Hill book company: New York, 1970

[Deák 1981] I. Deák, "An economical method for random number generation and a normal generator" COMPUTING, vol. 27, pp. 113-121, 1981.

[Dempster 1994] A.P. Dempster and E..Brown, "Probabilistic belief network models and modelling in context", proceedings of Adaptive Computing and Information Processing conference, London, Jan 1994.

[Der Kiureghian 1987] A. Der Kiureghian, M. Asce, Hong-Z. Lin, and Shyh-J. Hwang, "Second-order reliability approximations" Journal of Engineering Mechanics, vol. 113, no. 8, pp. 1208-1225, 1987.

[Díaz 1989] A. R. Díaz, "A strategy for optimal design of hierarchical systems using fuzzy sets" in The 1989 NSF Engineering Design Research Conference: At College of Engineering, University of Massachusetts, Amherst, June 1989, J. R. Dixon Ed., pp. 537-547.

[Dixon 1987] J. R. Dixon, A. Howe, P. R. Cohen, M. K. Simmons, "DOMINIC I: Progress toward domain independence in design by iterative redesign", Engineering With Computers, vol.2, no.3, pp. 137-145, 1987.

[Dong 1987a] W. Dong and C.H. Shah, "Vertex method for computing functions of fuzzy variables", Fuzzy Sets and Systems, vol.14, pp. 65-78, 1987.

[Dong 1987b] W. M. Dong and F. S. Wong, "Fuzzy weighted averages and implementation of the extension principle" Fuzzy Sets and Systems, vol. 21, pp. 183-199, 1987.

[Dubois 1986] D.Dubois and H.Prade, "Possibility theory: an approach to computerized processing of uncertainty", New York: Plenum Press, 1986.

[Fajdiga 1996] Fajdiga, M., Jurejevcic, T. and Kernc, J, "Reliability Prediction in Early Stages of Product Design", Journal of Engineering Design, Vol.7, No.2, pp. 107-128, 1996.

[Fuhr 1994] N. Fuhr and U. Pfeifer, "Probabilistic information retrieval as a combination of abstraction, inductive learning, and probabilistic assumptions" ACM Transactions on Information Systems, vol. 12, no. 1, pp. 92-115, January. 1994.

[Haugen 1980] E. B. Haugen, "Mechanical element optimisation" in Probabilistic Mechanical Design. New York: John Wiley & Sons, 1980, ch. 11, pp. 515-545.

[Hohenbichler 1981] M. Hohenbichler and R.Rackwitz, "Non-normal dependent vectors in structural safety", Journal of the Engineering Mechanics Division - ASCE, vol. 107, no. 6, pp. 1227-1238.

[Iman 1980] R. L. Iman, J. M. Davenport and D. K. Zeigler, "Latin hypercube sampling (program users guide)", prepared by Sandia Laboratories for the US Department of Energy under contract DE-AC04-76DP00789, SAND79-1473.

[Jokinen 1994] P. A. Jokinen, "Visualization of multivariate processes using principal component analysis and nonlinear inverse modelling" Decision Support Systems, vol. 11, pp. 53-65, 1994.

[Kaufmann 1985] A. Kaufmann and M. M. Gupta, "Introduction to fuzzy arithmetic". New York: Van Norstand Reinhold, 1985.

[Keeney 1976] R. L. Keeney and H. Raiffa, "Decisions with multiple objectives: preferences and value tradeoffs". New York: John Wiley & Sons, 1976.

[Kesselring 1951] F. Kesselring, "Bewertung von konstructionen, ein mittel zur steuerung von konstruktionsarbeit". Dusseldorf: VDI-Verlag, 1951.

[Kwaan 1994] J.R.Kwaan, "Bayesian belief networks for industrial applications", proceedings of Adaptive Computing and Information Processing conference, London, Jan 1994.

[Lehane 1990] K. J. Lehane, "Computer aids for variant design", Thesis, University of Bristol, 1990.

[Mardia 1979] K. V. Mardia, J. T. Kent, and J. M. Biddy, "Principal component analysis" in Multivariate Analysis. London: Academic Press Inc, 1979, pp. 213-254.

[McKay 1979] M. McKay D, W. Conover J, and R. Beckman J, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code" Technometrics, vol. 21, no. 2, pp. 239-245, 1979.

[McMahon 1993] C. A. McMahon, D.J. Pitt, J. Sims Williams H, and Y. Yong, "Review: An information management system for design data", Proceedings of the 1993 ASME Computers in Engineering Conference and Exposition, pp. 215-226, August. 1993.

[McMahon 1994] C. A. McMahon, "Observations on Modes of Incremental Change in Design", in press JED, 1994.

[Mileham 1993] A. R. Mileham, G. C. Currie, A. W. Miles, and D. T. Bradford, "A parametric approach to cost estimating at the conceptual stage of design", JED, vol. 4, no. 2, pp. 117-125, 1993.

[Moore 1966] R. E. Moore, "Interval analysis".Prentice-Hall: Englewood Cliffs, NJ, 1966.

[Moore 1979] R. E. Moore, "Methods and applications of interval analysis". Philadelphia: SIAM, 1979.

[Morgan 1990] M. Granger Morgan and M. Henrion, "Uncertainty, a guide to dealing with uncertainty in quantitative risk and policy analysis". Cambridge: Cambridge University Press, 1990.

[Nicklaus 1987] D.J Nicklaus, S.S.Tong, and C.J.Russo, "ENGINEOUS: A Knowledge Directed, Computer Aided Design Shell", proceedings of the third IEEE conference on artificial intelligence applications, Feb. 1987.

[Otto 1991] K. N. Otto and E. K. Antonsson, "Trade-off strategies in engineering design", Res Eng Des, vol. 3, pp. 87-103, 1991.

[Otto 1993] K. N. Otto and E. K. Antonsson, "Extensions to the taguchi method of product design", Journal of Mechanical Design, vol. 115, pp. 5-13, March. 1993.

[Otto 1994] Otto, K. N. and Antonsson, E. K., "Design parameter selection in the presence of noise", Research in Engineering Design, vol. 6, pp. 234-246, 1994.

[Pahl 1984] G.Pahl and W.Beitz, "Engineering design", London: The Design Council, 1984.

[Pearl 1987] J. Pearl, "Bayesian decision methods" in Encyclopaedia of Artificial Intelligence, vol. 1, S. C. Shapiro, Ed. New York: John Wiley & Sons, 1987, pp. 48-56.

[Peters 1992] T. J. Peters, "Encoding mechanical design features for recognition via neural nets", Res Eng Des, vol. 4, pp. 67-74, 1992.

[Phadke 1989] M. S. Phadke, "Quality Engineering Using Robust Design", London: Prentice-Hall International inc., 1989.

[Proban 1989] "Proban Version 2 Theory Manual", A.S. Veritas Research Report No: 89-2023.

[Quadrel 1993] R. W. Quadrel, R. F. Woodbury, S. J. Fenves, and S. N. Talukdar, "Controlling asynchronous team design environments by simulated annealing", Res Eng Des, vol. 5, pp. 88-104, 1993.

[Ramachandran 1992] N. Ramachandran, N. A. Langrana, L. I. Steinberg, and V. R. Jamalabad, "Initial design strategies for iterative design", Res Eng Des, vol. 4, pp. 159-169, 1992.

[Rice 1988] "Fatigue Design Handbook", ed. Richard C. Rice, pub. Society Automotive Engineers Inc., Warrendale PA USA, ISBN 0-89883-011-7, pp 120 - 136, 1988.

[Robertson 1976] S. E. Robertson and K. Sparck Jones, "Relevance weighting of search terms", Journal of the American Society for Information Science, vol. 27, no. 3, pp. 129-146, 1976.

[Savoy 1994] J. Savoy, "A learning scheme for information retrieval in hypertext", Information Processing and Management, vol. 30, no. 4, pp. 515-533, 1994.

[Siddall 1983] J.N.Siddal, "Probabilistic engineering design: principles and applications". New York: Marcel Dekker inc., 1983.

[Smets 1988] P. Smets, P. H. Mamdani, D. Dubois, H. Prade, "Non-standard logics for automated reasoning". London: Academic Press, 1988.

[Sparck 1972] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval", Journal of Documentation, vol. 28, no. 1, pp. 11-21, 1972.

[Stone 1989] J. R. Stone, D. I. Blockley, and B. W. Pilsworth, "Towards machine learning from case histories", Civil Engineering Systems, vol. 6, no. 3, pp. 129-135, 1989.

[Taguchi 1986] G. Taguchi, "Introduction to quality engineering". White Plains, NY: Asian Productivity Organization, 1986.

[Tee 1991] A. B. Tee and M. D. Bowman, "Bridge condition assessment using fuzzy weighted averages", Civil Engineering Systems, pp. 49-57, 1991.

[Thurston 1991a] D. L. Thurston, "A formal method for subjective design evaluation with multiple attributes", Res Eng Des, vol. 3, pp. 105-122, 1991.

[Thurston 1991b] D. Thurston, "Design evaluation of multiple attributes under uncertainty", SARA, vol. 1, pp. 143-159, 1991.

[Thurston 1993] D. L. Thurston and A. Locascio, "Concurrent optimal design with application to structural dynamics", JED, vol. 4, no. 4, pp. 353-369, 1993.

[Tsai 1993] S. C. Tsai and K. M. Ragsdell, "Optimization efficiency of taguchi-class experimental design using conjugate arrays", JED, vol. 4, no. 3, pp. 167-188, 1993.

[Ullman 1992] Ullan, DG, "The Mechanical Design Process", New York : McGraw-Hill, ISBN 0-07-112871-9, 1992.

[Varghese 1996] Varghese, P., Braswell, R. N., Wang, B. and Zhang, C. "Statistical Tolerance Analysis using FRPDF and numerical convolution", Computer-Aided Design, vol. 28, no.9, pp. 723-732, 1996

[Vogwell 1990] J. Vogwell, "Computer-aided component selection: a new and expanding research activity", Computer-Aided Design, vol. 22, no. 5, pp. 308-310, 1990.

[Vogwell 1992] J. Vogwell, "A methodology for computerizing the selection of catalogue components", JED, vol. 3, no. 2, pp. 167-179, 1992.

[1993a] Ward, A. C. and Seering, W. P., "The Performance of a Mechanical Design 'Compiler'", *Journal of Mechanical Design*, vol. 115, pp. 341-345, Sept. 1993.

[1993b] Ward, A. C. and Seering, W. P., "Quantitative Inference in a Mechanical Design 'Compiler'", *Journal of Mechanical Design*, vol. 115, pp. 29-35, Sept. 1993.

[Wong 1993] A. Wong and D. Sriram, "SHARED: An information model for cooperative product development", Res Eng Des, vol. 5, pp. 21-39, 1993.

[Wood 1989] K. L. Wood and E. K. Antonsson, "Computations with Imprecise Parameters in Engineering Design: Background and Theory", Transactions of the ASME, vol. 111, pp. 616 - 625, Dec. 1989.

[Wood 1990] K. L. Wood, E. K. Antonsson, and J. L. Beck, "Representing imprecision in engineering design: comparing fuzzy and probability calculus", Res Eng Des, vol. 1, pp. 187-203, 1990.

[Wood 1994], T. L. Wood, "Fuzzy logic techniques applied to the automated selection of ill-defined standard components", Thesis, University of Bath, 1994.

[Zadeh 1975] L. A. Zadeh, "The concept of linguistic variable and its application to approximate reasoning", Information Science, vol. 8, 1975.

[Zangemeister 1970] C. Zangemeister, "Nutwertanalyse in der Systemtechnik" . Munich: Wittemannsche Buchhandlung, 1970.

Appendix C: Transcript of Part of Facia Workshop Held at Warwick University on 12th September '94

Present:

Rose Crossland	(Bristol University)
Neil Davis	(Warwick University)
Ed Jackson	(Team Leader for Trim and Hardware Interior, Design, Rover)
Lorna MaCaulay	(Team Leader for Trim and Hardware Interior, Purchasing, Rover)
John Raulston	(Vehicle Cost Estimator, Rover)
Dave Simmonds	(PSIS, Rover)
Jon Sims Williams	(Bristol University)

ND: Is that an area of uncertainty that is significant, undercosting?

JR: I don't believe a lot of that is done deliberately to be honest, its done through sheer error. Even the number of components. Because we tend to be focusing on a Pareto base - looking at the big items - the facias, the bumpers, the engine,... we tend to miss a lot of the smaller bits. When you actually get a vehicle cost office tabulation you think, we're missing half of it. And they add up to 2% or 3% under-costing.

ND: What techniques do engineers use to get around that? I mean no-one expects you to get a cost for every item on the vehicle.

JR: I should be interested to hear the answer to that to be honest. We've traditionally left the non-key elements to finance because they've got an up to date tabulation of what the last vehicle (the replacement vehicle) cost. In terms of ...for all the VPGs there's an element of fixings for instance to take a simplistic view. They know that by and large, those fixings are going to be much the same for the next one, and so on.

EJ: That's the way it has happened historically. There's global fixing elements within VPGs. We took this project further, when we did our cost sheet and we broke it down into fixing items per component assembly. That's evolved over a period of time and illustrated parts lists helped us to look for the fixings. As John says, 2% to 3% of the cost is quite significant. So we're actually creating a spec for the product now, before we get to D0. So it is getting better, by refinement.

.

.

.

ND: What about risk? What does risk mean to you and where does it fit into this process and where does it fit into the D0 event?

LM: It is identified at D0 generally. We have a feature and a cost which have some levels of opportunity and some levels of risk identified.

JR: Yes, but they're financially evaluated

LM: Generally financially, yes

JR: yes but I think there's other factors to that, such as the <inaudible> risk to design which are not evaluated financially, such as crash-worthiness and so on.

ND: So, there's a certain amount of pounds risk. What were you saying about that Lorna?

JR: Well ECM [Effective Cost Management Guidelines] dictates that you set an agreed target figure at D0 which is an accumulation of risk and opportunity basically.

ND: Risk meaning overspend and opportunity meaning under-spend?

JR: Well, no, I mean, the risk is the evaluation of what can go wrong. In terms of the easily identifiable bits. For instance, should we need to paint or not paint. This sort of thing. The simple elements.

LM: But surely that's identified as an opportunity, not to paint if there's an extra cost or whatever?

JR: Well, yes, but it's also a risk.

LM: Yes, I'm saying, risk or an opportunity, either way. We should identify those at D0.

ND: Are they statements like "we may have to paint or we may not have to paint" or do they translate into the fact that "if we have to paint then its going to be £27.60 if we don't then its going to be £27.50".

JR and LM: Yes (the latter)

ND: So there are actually cost values against these risks?

JR: Yes, and they're applied.

EJ: But there's also things like economics, raw materials supply, they come into the risk.

LM: Yes, as long as you highlight them with as much level of confidence as you can.

JR: Economics is a thing of the past in that respect.

LM: Not if they're minus John.

JR: Well, that's not a risk, that's an opportunity.

ND: I've been told by finance that the vehicle team shouldn't worry about economics because that's a group function. But is that something that teams like to do anyway?

LM: To be honest, we've never really spoken about it much in core reviews. It's generally been that you talk around the feature and then the economics is then generally dealt with by purchasing on a yearly basis. What we try and do is get a level of stability throughout. What we're looking at this year is having 3, 4 or 5 year contracts or as loose-term contracts as we can where suppliers are expected to support year-on-year reductions, on the basis of obtaining more business. We try and keep those away from target costs because its very difficult to consolidate somebody taking a blanket figure off their bottom line and giving somebody an accurate target cost for a component. So there's a lot of discussion going on along those lines as to how we can justify doing what we're doing, bearing in mind we operate an ECM [Effective Cost Management] process. So it's quite difficult.

ND: If we look at the pounds thing on a piece part cost, for example, who actually evaluates that risk, who comes up with the figures?

JR: We do it jointly with the supplier.

ND: Right, OK. So are we saying that for every identified part at D0 there is a min-max?

JR: No, its not a min-max, its a balance of that is fed into what we see as a target. The target is bought off. I think it doesn't do because there's only a known performance of that particular product at that time. For instance, it wouldn't take into account Ed's facia falling into pieces at the first impact. This sort of thing. You can't do that until you've either got a product, a prototype to try it on. You can't build that sort of risk in. It has happened on one product, it caused us major heartache.

EJ: You try to eliminate that risk through your design processes, don't you.

JR: Yes, you do but there's some things you .. where the cost of guarding against it outweighs the costs of feeding it in as a risk in the first place.

ND: I haven't quite understood this because you've said that the risk is taken into account at D0.

LM: As much as possible. As much as we can identify. Bearing in mind John did say that we don't actually capture, or we maybe capture on a Pareto basis the number of components we do. So there's always the ones that miss.

ND: Would you always have some risk allocated to the area then? Or is just a matter of what we can calculate we calculate and what we can't we won't?

JR: There's not actually a sub-total of all the risks, for instance, to Ed's area. So far as I'm aware there's not is there Ed?

EJ: We don't need confidence levels...

ND: What do you mean by confidence levels?

3 **EJ:** If they're a supplier +/-5%, engineer +/- 15%, VCE [Vehicle Cost Estimator] +/- 10%. So depending on where the information had come from, there are different levels of confidence. But of course, by the time we get to D0 we should have a supplier quote anyway.

JR: That's not risk in terms of performance. Its confidence levels.

EJ: We're talking risk in terms of pounds, not performance.

JR: Maybe it's me that's got this crossed then, but I was looking at risk in terms of does the product work, if not we have to this, therefore the financial risk is X. Not, it's over-valued or under-valued.

LM: Well there's lots of different types isn't there. As long as we have the ability to cover them all then we're safe.

ND: Is confidence something that you see as being different to risk or partly wrapped up in the risk?

EJ: I see them differently. Confidence is very much against the price given for a known entity. Risk is against the unknown - the fact that you failed to meet performance or your tool doesn't operate. Things like that.

ND: Is it fair to say that, in your terminology, that risk is more to do with the physical performance of the parts of the vehicle or the cost performance?

EJ: I know what I'd say as an engineer.

LM: Well, we'll all say different things won't we. But because we work in a team - the idea is to have an engineer, purchasing, VCE and supplier in a team - we get a mixture of everybody's views and so long as you identify them within that environment then generally you manage to catch as much as you can.

ND: But not all of those risks that you identify you can quantify?

EJ: Yes. I agree. We certainly haven't been able to but..

JS: What will you do with this risk or confidence that you have? I can see that you will talk around it and I can see that there's a decision made on things like supplier or what particular configuration, that's probably not the right word, for your facia you decide on, will be influenced by your perception of risk.

EJ: You try to reduce the risk by the processes you use. Process for design, reliability predictions, mould flow analysis, finite element analysis. We do all those sort of things to try and minimise it but there's still the physical quantity that you need to test at the end of the day. So you've reduced it to what you would say is an acceptable level of risk, otherwise you put your hands up and say "I've got a problem".

ND: I've certainly heard people talking about rolling up cost. That sounds fairly intuitive about trying to get to some vehicle cost by rolling up the individual costs. Do you do the same for risk? Or is it more a case of - here's a great long list of the risks and we'll just evaluate them subjectively.

EJ: Risks usually aren't cumulative are they? In my experience.

ND: So they're absolute risks?

EJ: Yes, specific items. The task is to analyse the risk to get to an acceptable confidence. But also, in my view I like to have a contingency. If there is a contingency that it's going to go wrong, what am I going to do about it? At that point in time, rather than wait to get there and find that it has gone wrong, how do I leave myself room to manoeuvre out of that situation.

ND: In an environment where people can see your spend, there's a risk then that your contingency is taken away from you. What I'm trying to get to is at what level, at what stage, can risk evaluations be brought onto the engineer's desktop, so that the contingency is built into the data rather than being separate?

4 **JR:** D0, it comes in at D0. We'll feed it in through the CDKS process anyway, through ECM. Risk and opportunity will both be evaluated and balanced. We will presumably have a table of experts around the table who can evaluate risk. People from the manufacturing, from our area, from engineering etc. we'll all put our heads together and say well the risk is real. If we think it's real, we feed it in and it becomes part of the target. The risk is fed in, its not a bottom-line contingency, it must be identified as Ed said, by every component.

ND: So come the D0 event people are basically voting on it are they?

LM: No, it's the result of many months of discussion and re-iteration.

JR: It's never shown as a block item at the bottom. But it should be there on every component.

LM: There have been situations when I first started on Landrover projects where they said to us we'd like to sign off a target cost for a component and actually list out what your opportunities are or what you think might happen if we go this route. And try to put a cost against it. We did actually do that for a while and then it sort of fell into disrepute. Because I don't think we did it properly - I certainly didn't have the knowledge to work out the current cost <inaudible> or whatever. But we did run that for a while on CCS, piggy-backed, and then it sort of faded away. I don't know whether anybody else has had that, certainly I did for a while and that was it then.

JR: Well we did it on for instance CB40 facia. We did the risk <inaudible>

ND: But do people say, well what if all our worst nightmares come true, and take all the risk values and tot them all up and say OK, well that's the most risky scenario. And do people do the same for all the opportunities and say, what happens if all the opportunities come in. To give you an overall thing that says, OK, the capital cost of this vehicle is going to be somewhere between 200 and 300 million pounds, and do they do the same for piece part costs, to say that the material cost is going to be between, say, 600 and 650 pounds? Does that get done?

JR: No. There's not a tolerance to an overall cost. It's never published.

ND: So there's some sort of subjective..., people around the table, saying "It's OK"

JR: Yes. Basically.

EJ: Isn't there an allocation made by the finance team?

LM: Yes, there is, but as John said its based on the past model cost or the past similar set of components or whatever. So yes, they do put something in.

JR: We should leave bottom line contingency, for instance, to one body. Not everybody should be involved. Let the guys who add the numbers up at the end of the day and are responsible for the financial performance of the company, or the monitoring of, do the contingency work.

ND: But they won't know the technical implications will they?

JR: No, they're not technical. Those are taken care of in the target costing. We're talking about a lot of external factors such as economics..

LM: Price of raw materials..

ND: No, I understand that. But if..

JR: Currency economics etc. We're not talking about engineering risk now, we're talking about contingency. That should be catered for within the target cost.

ND: Yes, but at some stage, you as vehicle programs or as a team leader, hand over your information to the finance person and say "facia on CB40 is going to be X and Y". Do you say, its going to be between this and this for piece part and this and this for tooling, or are you just saying a single value?

EJ: I suppose I do indirectly, because I'll say what the LIC [Latest Indicated Cost] is and then I've got a target as well which is always below the LIC. So by engineering detail you try to achieve that target.

LM: But your target is made up of all your risks and opportunities that you identified at D0. That's where the target comes from, its something that's bought off between everybody that we know is achievable. Having recognised that there are opportunities and risks with that component.

JR: The onus is not on just one body to achieve that target. The onus is on our engineering as well as the supplier.

5 **ND:** So are we saying that at D0, we've got, for facia lets say, a risk and opportunity for facia. That the overall programme risk is only evaluated based on the most likely value that's accumulated, and then at the programme level the economics and the currency fluctuations are then taken into account. But nevertheless, you've still got your plus and minuses down at your level. Those aren't visible to the total programme, but they are then moved forward..

JR: I doubt if those risks and opportunities are visible from a team leader point of view.

LM: Unless they get involved in looking at CDTSs [document signed by supplier prior to D0 which commits supplier to cost target] then ..

EJ: Not all of them..

JR: [to EJ] You have a great list of them don't you..

ND: So risks are identified but not communicated ..

LM: No, they are communicated within the working team. Within the people who will have direct effects on those costs. As in the buyer, the engineer, the vehicle cost man and the supplier. We'll all have a view at that level. But whether Ed chooses to get involved in monitoring all the CDTSs, is really up to him. But they are available should he require to see them. I think you're in danger then of having so much paperwork flying around that not a lot else of work is done. As long as its available to the people who actually do the work and are empowered to make those decisions, then I think that's all that's necessary.

JS: I think the big question we're asking is supposing that a system automatically looked at this for you, so that all you had to say was, when you're doing this little bit, this is the range of costs we're expecting and this is the range of time delays that may be involved, in tooling or something, and all of that data you just put in as you were doing that thing, and some program was just looking at the whole lot and saying, well lets have a look at all these things and see if they're going wildly wrong, would that be useful or not?

JR: Not really

EJ: My first reaction is I don't think so. I find a range is so open to interpretation that it depends on the detail of the products that you're designing.

LM: We're trying to get away from ranging costs as well. We're trying to reach the stage where we can pinpoint fairly early on in the program how much something is going to cost. It's nice to have some things to look at but really because the organisation is going leaner, there's less people around, to do that analysis, we're trying to get smarter by having a knowledge of what a specific item costs rather than a range of ..

JR: After all, a cost is a fact.

LM: Yes.

JS: Well a cost isn't a fact actually is it until you've signed a piece of paper for it.

JR: Well its still a fact, whether you've signed for it or not. It's not an estimate.

ND: But John, you're doing estimates aren't you. Up until you've actually signed a contract, as Jon said, its an estimate, its an expectation.

JR: Yes

ND: Until the supplier actually says, ..

JR: But we're talking now of a supplier who's loosely contractually bound by that target cost. He's put his signature to it. If he doesn't, fair enough there probably isn't a legally binding contract there, but if he doesn't he knows that future business coming his way will be in jeopardy.

ND: So these are signatures for QAFs [Quotation Analysis Forms]?

LM: CDTSs Everyone signs them.

6 **JS:** So the implication is that Rover and the supplier are basically into this partnership. And if something goes wrong, then you're both going to have to bear a bit of the aggro?

JR: If the supplier has got an extraordinary cost, that on examination he cannot get rid of then the onus has got to fall back on Engineering to try and get rid of it by another means. And the engineer has been given responsibility for cost because theoretically he is the only one empowered to change it. Because he's the only one that can actually change the functionality, the design, etc. None of us can actually change that.

LM: Unless all we do is lop something off their margin by asking for a blanket price. But it doesn't do any good.

JR: Because that puts him in jeopardy and he's not a good supplier any more.

Appendix D: The Base Classes

D.1 Introduction and notation

The complete inheritance hierarchy for the base classes is shown in Fusion object model notation in Figure D-1, Figure D-2 , Figure D-3 and Figure D-4. All the classes shown in these diagrams are described in this Appendix. For each class there is a textual description, and a Fusion class definition. The class definitions are reproduced from the schema file - see Appendix E for an explanation of the class definition notation.

For each method belonging to the class, the attribute values which it uses are listed immediately below the class definition. In general, if any attribute values used by a method are UNKNOWN or if any links used by the method are NONE or UNKNOWN then the method will return a value of UNKNOWN. If a method can still return a value when a link has a value of NONE, then the link name is shown enclosed in round brackets, (thus). Here is an example taken from the `Assembly` class definition:

```
piece_cost_fn() uses:
        components.piece_cost
        (assembly_process).process_cost
        (assembly_process).material_cost
```

This indicates that the method `piece_cost_fn()` will still return a value if there is no assembly process defined for the assembly, but will not return a value (i.e. will return a value of UNKNOWN) if the assembly has no components.

D.2 Inheritance diagrams

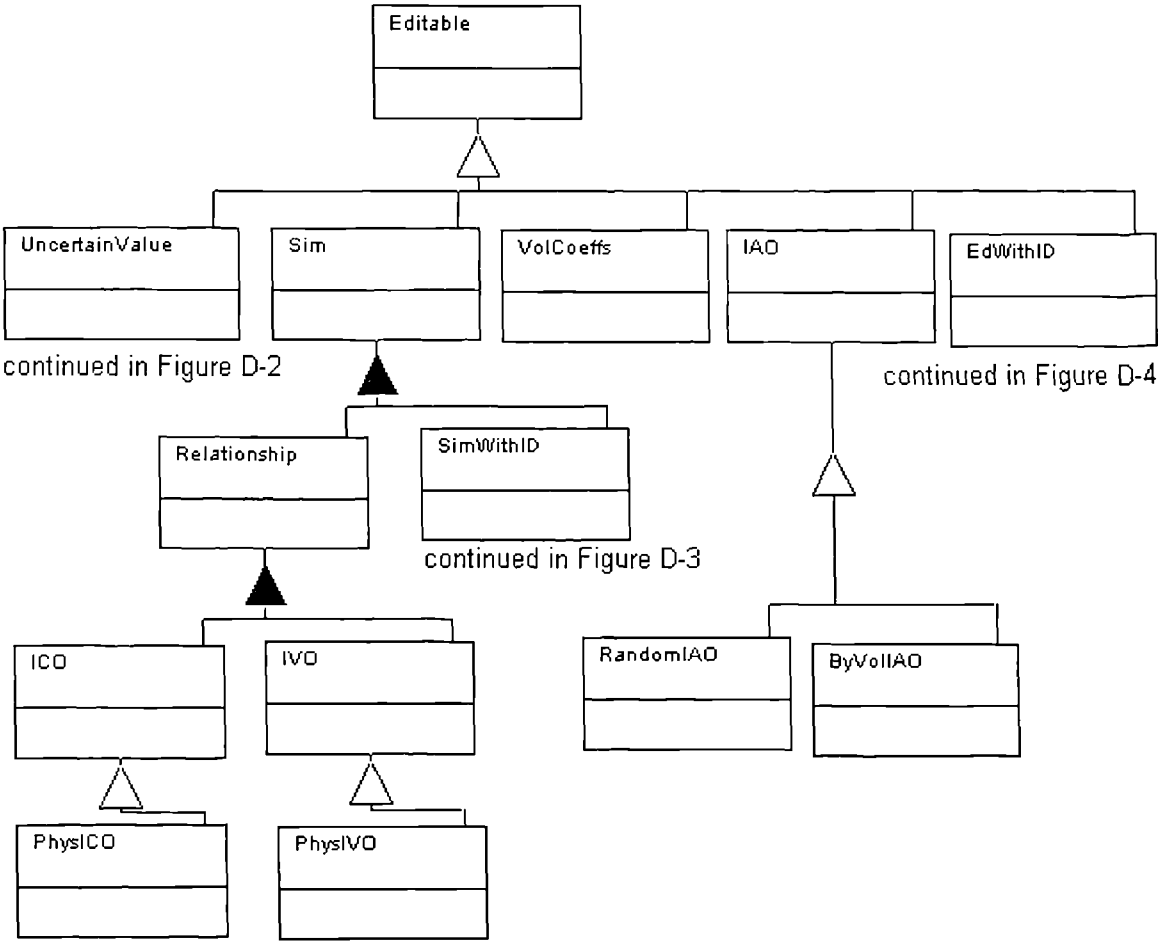


Figure D-1: Top level view of base class inheritance hierarchy

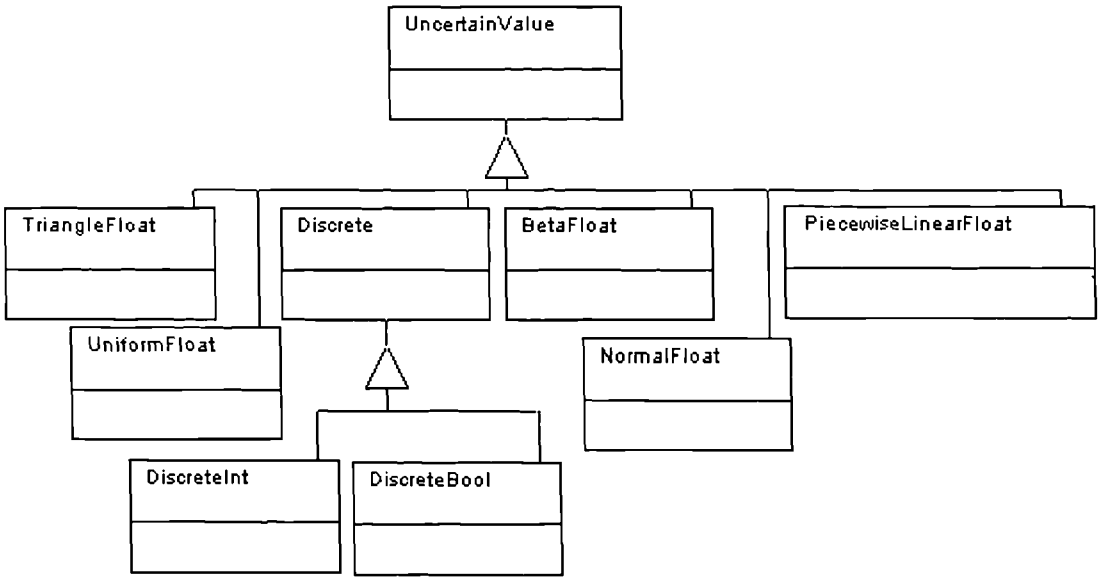


Figure D-2:UncertainValue inheritance hierarchy

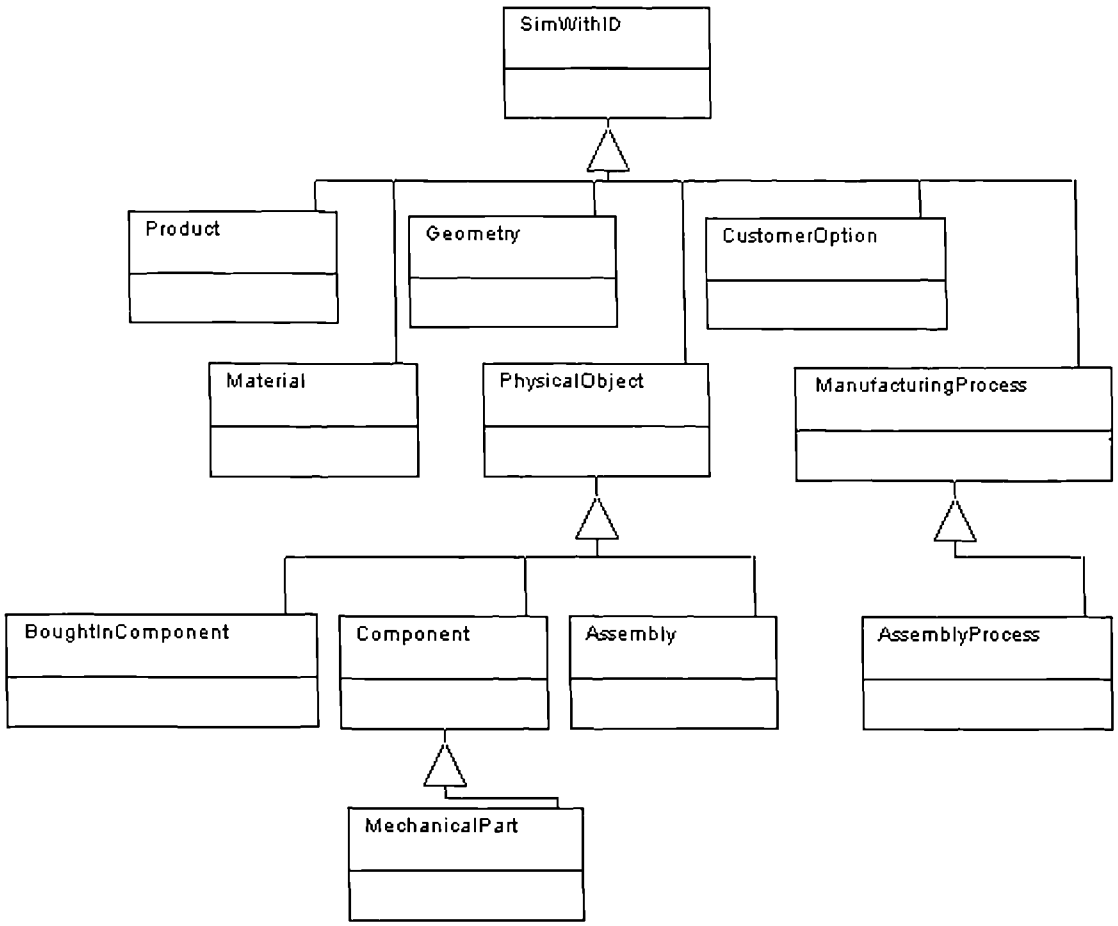


Figure D-3: SimWithID inheritance hierarchy

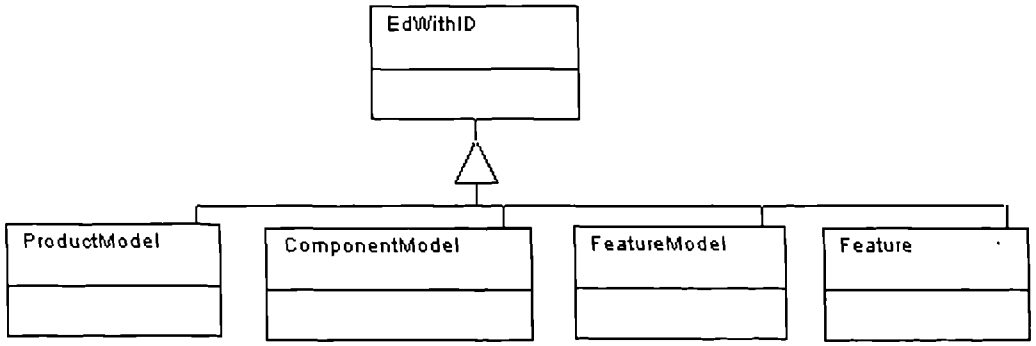


Figure D-4: EdWithID inheritance hierarchy

D.3 Editable and Sim

```
class Editable isa
endclass
```

The Editable class provides object persistence and visibility in RiTo. `Editable` is the root class of the RiTo base classes - thus all RiTo object classes are derived directly or indirectly from `Editable`.

```
class Sim isa Editable
endclass
```

An instance of `Sim` (a "Sim object") is an object in a risk model representing part of a design. A `Sim` object may have uncertain attribute values, alternative link values and may have multiple derivation routes defined for any of its attributes.

D.4 EdWithID and SimWithID

```
class EdWithID isa Editable
 attribute variable name : Str
 attribute variable description : Str
 attribute variable identifier : INTEGER
endclass

class SimWithID isa Sim
  attribute variable name : Str
  attribute variable description : Str
  attribute variable identifier : INTEGER
endclass
```

The name of a `SimWithID` or `EdWithID` object is displayed by RiTo when browsing the model. Description and identifier fields are provided for user information.

D.5 IAO and derived classes

D.5.1 IAO

A risk model may contain several alternatives for a particular `Sim` object. For example, there may be several alternative assembly processes under consideration for an assembly. Or there may be two different designs, A and B, for the whole assembly under consideration. In this case an `IAO` (is-alternative-of) object is used to represent the relationship between A and B. Two kinds of `IAO` object are provided in the base classes: `RandomIAO` and `ByVolIAO`. In a `RandomIAO`, a probability is assigned to each alternative. This represents the perceived likelihood of that alternative eventually being chosen. By contrast, in a `ByVolIAO`, the choice is not an independent random variable, but depends upon another variable in the model. Here, the choice made between the alternatives is determined by the production volume of a specified `PhysicalObject`.

An instance of an `IAO` class stores the class of the objects which it relates - it may relate objects of this, or a derived, class. This is termed the `IAO` parameter class (because the class itself is a parameter of the `IAO`). Links may then be defined to the `IAO` object from elsewhere in the model - provided that the `IAO` parameter class is the same as (or derived from) the class of the link.

The `IAO` class (which relates alternative objects, only one of which will eventually be manufactured) should not be confused with the `IVO` class, which relates a set of variant objects which will all be manufactured, but only one of which will be purchased by a particular customer.

```
class IAO isa Editable
 attribute variable alternatives : unbound Editable[]
 attribute variable selection : unbound Editable
endclass
```

The alternatives are links to each of the alternatives under consideration. Selection is a link to the selected alternative. Objects in the risk model are usually instances of classes derived from `IAO`, rather than direct instances of `IAO` itself, since `IAO` provides no methods for making a selection between the alternatives.

D.5.2 RandomIAO

```
class RandomIAO isa IAO
 attribute constant probabilities : bound Discrete
 method choose_random : Editable
endclass
```

`Probabilities` is a link to an `UncertainValue` object representing a discrete probability distribution, containing the probability of each alternative. The discrete values taken by this distribution are the indexes into `alternatives` (inherited from `IAO`). The `choose_random()` method takes a random sample from the discrete probability distribution and returns a pointer to the alternative which it indexes.

Instances of `RandomIAO` will have the `choose_random()` method assigned as the derivation route for selection. This (`IAO::selection`) is the only link in the model which is permitted to have a method defined as its derivation route. And `IAO` methods (such as `choose_random()`) are the only methods whose return type may be an object, rather than one of INTEGER, BOOL or FLOAT.

D.5.3 ByVolIAO

```
class ByVolIAO isa IAO
 attribute constant physical_object : unbound PhysicalObject
 attribute variable prod_vols : FLOAT[]
 method choose_by_prod_vol : Editable
endclass
```

This class is used in a situation where the choice made between alternatives will be determined by the production volume of `physical_object`. The attribute `prod_vols` defines the range of production volumes for which each alternative will be chosen. Instances of `ByVolIAO` will have the `choose_by_prod_vol()` method assigned as the derivation route for selection. The attribute `prod_vols` must have a cardinality which is one less than the number of alternatives.

If there are *N* alternatives, the `choose_by_prod_vol()` method makes its selection as shown in the table below. `Alternatives[`**Index**`]` is chosen if `physical_object.prod_vol` lies in the interval (**Lower**, **Upper**] where:

Index	**Lower**	**Upper**
0	-infinity	`prod_vols[0]`
1	`prod_vols[0]`	`prod_vols[1]`
...	...	...
N - 2	`prod_vols[N - 1]`	`prod_vols[N - 2]`
N - 1	`prod_vols[N - 2]`	+infinity

D.6 UncertainValue and derived classes

`UncertainValue` objects represent probability distributions. The attributes of an `UncertainValue` object are the parameters of the distribution. See Figure D-2 for the inheritance hierarchy of `UncertainValue` objects.

D.6.1 TriangleFloat

`TriangleFloat` represents a triangular distribution and generates samples of type FLOAT:

```
class TriangleFloat isa UncertainValue
 attribute variable min : FLOAT
 attribute variable likely : FLOAT
 attribute variable max : FLOAT
endclass
```

D.6.2 Discrete, DiscreteInt and DiscreteBool

`Discrete` represents any discrete random variable and generates a sample which is an index into a variable cardinality attribute (or link in the case of a `RandomIAO`). It is only directly instantiated when it is part of a `RandomIAO` - otherwise, derived classes are used.

```
class Discrete isa UncertainValue
 attribute variable probs : FLOAT[]
endclass
```

`DiscreteInt` represents a discrete integer and generates a sample which is of type INTEGER and is one of the values stored in `vals`. In an instance of `DiscreteInt`, the cardinality of `vals` must be the same as the cardinality of the inherited attribute `probs`. `probs[i]` is the probability of the discrete integer taking a value of `vals[i]`.

```
class DiscreteInt isa Discrete
 attribute variable vals : INTEGER[]
endclass
```

`DiscreteBool` represents a random BOOLEAN variable which may take values of TRUE or FALSE. It generates a sample of type BOOL. The cardinality of the inherited attribute `probs` must be 2. The value of `probs[0]` is the probability of a value FALSE and the value of the inherited attribute `probs[1]` is the probability of a value TRUE.

```
class DiscreteBool isa Discrete
endclass
```

D.6.3 PiecewiseLinearFloat

`PiecewiseLinearFloat` represents any distribution which can be sketched as a series of points joined by straight lines and generates samples of type FLOAT. Each point is a <probability, value> pair and is stored as `probs[i]` and `vals[i]` for some i. `Probs` and `vals` must both have the same cardinality. Zero-probability intervals are permitted. RiTo will normalise the probability values - it is only their relative size (ratio) which is important.

```
class PiecewiseLinearFloat isa UncertainValue
 attribute variable probs : FLOAT[]
 attribute variable vals : FLOAT[]
endclass
```

For example, the distribution illustrated in Figure D-5 could be represented in an object repository file as

```
Object  : 16
Class   : PiecewiseLinearFloat
FLOAT probs = 1, 0, 0, 2, 3, 0
FLOAT vals =  1, 2, 3, 3, 6, 9
```

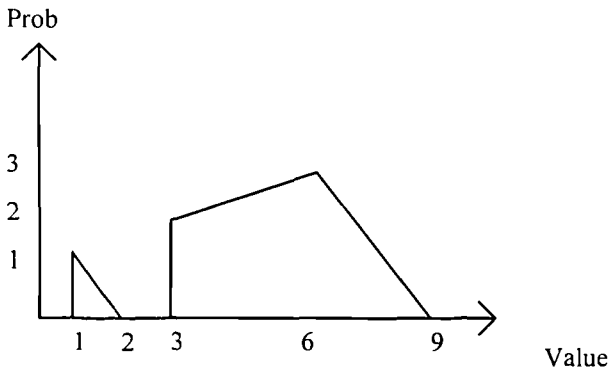


Figure D-5: Example piece-wise linear distribution

D.6.4 UniformFloat

`UniformFloat` represents a uniform (rectangular) distribution and generates samples of type FLOAT. It should be used to represent attributes where only the interval of possible values is known.

```
class UniformFloat isa UncertainValue
 attribute variable min : FLOAT
 attribute variable max : FLOAT
endclass
```

D.6.5 NormalFloat

`NormalFloat` represents a normal distribution centred about `mean` with a standard deviation of `sd` and generates samples of type FLOAT.

```
class NormalFloat isa UncertainValue
 attribute variable mean : FLOAT
 attribute variable sd : FLOAT
endclass
```

D.6.6 BetaFloat

`BetaFloat` represents a beta distribution with first parameter (sometimes termed alpha or a) of a1 and second parameter (sometimes termed beta or b) b2. It generates samples of type FLOAT.

```
class BetaFloat isa UncertainValue
 attribute variable a1 : FLOAT
 attribute variable a2 : FLOAT
endclass
```

D.7 Overview of components and assemblies

The entity-relationship diagram in Figure D-6 indicates the basic structure of the classes. This diagram explains the distinction made between an assembly and a component - a component cannot be further decomposed and has a single material, whereas an assembly may have many components - each made of a different material. A part with more than one material should be modelled as an assembly - unless the additional material can be regarded as a part of the assembly process, for example when painting a door panel. The exception to this rule is bought-in components - any item which is purchased from a supplier as a single entity is modelled as a bought-in component, regardless of its physical make-up.

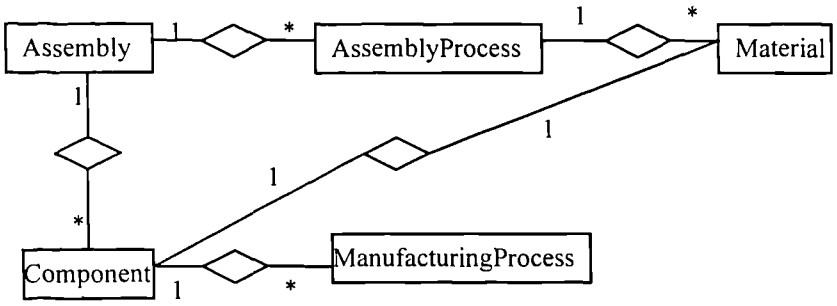


Figure D-6: Entity-relationship diagram for components and assemblies

Mechanical parts also have a geometry (as shown in Figure D-7), in addition to a material and zero or more manufacturing processes. Note that this structure does not preclude attaching geometry and process information to assemblies before they have been decomposed - consider the example of an armature moulding which consists of a PVC carcass, a skin and a foam filling between the skin and the carcass. Initially we only want to model the basic carcass with its material, manufacturing process and geometry. The armature moulding is modelled as an assembly with an assembly process of NONE and a single component (a mechanical part, the carcass). Later, a second component (the skin) is added - again with its own material, manufacturing process and geometry. Later still, an assembly process is added which consists of injecting the foam and has the foam as its material.

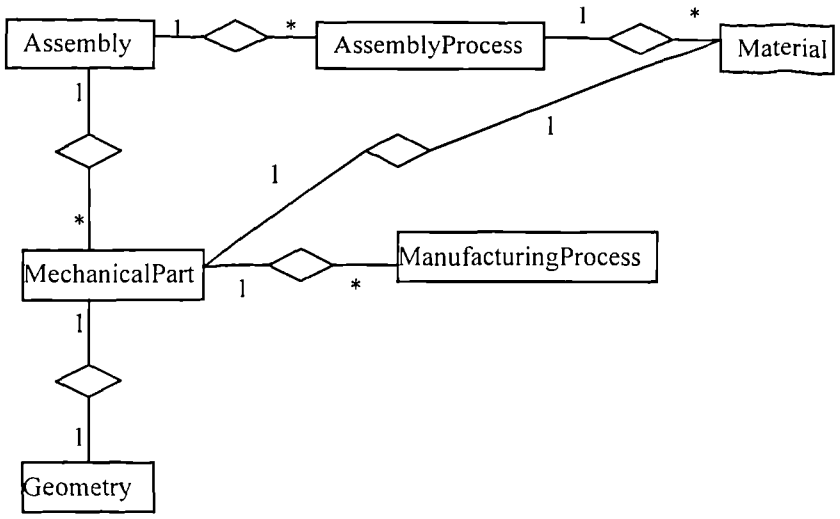


Figure D-7: Entity-relationship diagram for mechanical parts

In the following sections, each of these classes is described.

D.8 PhysicalObject and derived classes

D.8.1 PhysicalObject

```
class PhysicalObject isa SimWithID
 attribute variable piece_cost : FLOAT
 attribute variable tool_cost : FLOAT
 attribute variable logistics_cost : FLOAT
 attribute variable prod_vol : INTEGER
 attribute variable vol_coeffs : bound VolCoeffs
 attribute variable variants : bound PhysIVO

 method prod_vol_fn ( ) : INTEGER
 method piece_cost_fn ( ) : FLOAT
 method tool_cost_fn ( ) : FLOAT
 method logistics_cost_fn ( ) : FLOAT
endclass
```

```
prod_vol_fn() uses:
        feature_model.product[].sales_volume
        vol_coeffs.per_product[]
        vol_coeffs.extra_per_option[]
        vol_coeffs.(affected_by_option[]).take_up_percentage
```

The piece cost is the cost incurred for each physical object which is manufactured - generally made up of material costs, manufacturing processing costs, sub-component costs and so on. The logistics cost is again incurred for each physical object which is manufactured - this is the cost associated with packaging and transporting the object. The tool cost is the total cost associated with preparing to manufacture the object (e.g. providing machine tools, assembly stations etc.) - and thus is only incurred once regardless of how many of the objects are manufactured.

The attribute called `prod_vol` represents the production volume for this `PhysicalObject`. The sub-object called `vol_coeffs` stores coefficients which are then used by the `prod_vol_fn()` method to calculate the production volume. `Vol_coeffs` and `variants` are generally only used for *Sim* models which include variants.

The link called `variants` should only have a value if this `PhysicalObject` is a generic object - i.e. it has a set of variants. For an ordinary `PhysicalObject`, the value of `variants` should be NONE. This link should never be assigned an `IAO` or UNKNOWN value - it should always be either a point value or NONE. If a `PhysicalObject` is a generic object, then its piece, tool and logistics costs should be obtained using the methods provided in the `IVO`.

The piece, tool and logistics cost methods in `PhysicalObject` are defined in sub-classes - if `PhysicalObject` is instantiated directly and its methods are invoked they will return a value of UNKNOWN.

The `prod_vol_fn()` method calculates the production volume for this physical object using the stored production volume coefficients along with the estimated sales volumes for all the products this physical object is used in and all the customer options it is used in. This is achieved by calculating the sum over all products in the feature model of:

```
product[i].sales_volume *
    (
     vol_coeffs.per_product[i] +
     vol_coeffs.extra_per_option[i] * vol_coeffs.affected_by_option[i].take_up_percentage
    )
```

D.8.2 BoughtInComponent

```
class BoughtInComponent isa PhysicalObject
 attribute variable supplier : Str
 attribute variable quote_matl_cost : FLOAT
 attribute variable quote_proc_cost : FLOAT
 attribute variable quote_tool_cost : FLOAT
 attribute variable quote_logistics_cost : FLOAT
 attribute variable quote_piece_cost : FLOAT

 method quote_piece_cost_fn : FLOAT
endclass

quote_piece_cost_fn() uses:
        quote_matl_cost
        quote_proc_cost
```

Bought-in components are components where responsibility for calculating or estimating the costs has been delegated entirely to the supplier and the risk model need only include the (possibly uncertain) quotes given by the supplier. An example would be a sound system for a car. The automotive manufacturer would be unlikely to attempt to model the manufacturing process etc. for such an item.

Attributes are provided to store the material, processing, tooling and logistics costs quoted by the supplier as well as the name of the supplier. The `quote_piece_cost_fn()` method calculates the `quote_piece_cost` by adding the quoted material and process costs. Typically, the preferred derivation route for the inherited attributes of `piece_cost`, `tool_cost` and `logistics_cost` would be from the quoted values, with an in-house estimate as a lower priority route to be used before quotes were obtained.

D.8.3 Component

```
class Component isa PhysicalObject
 attribute variable material : unbound Material
 attribute variable manufacturing_process : bound ManufacturingProcess[]

 method piece_cost_fn : FLOAT
 method tool_cost_fn : FLOAT
endclass

piece_cost_fn() uses:
        manufacturing_process.material_cost
        manufacturing_process.processing_cost


tool_cost_fn() uses:
        manufacturing_process.tool_cost
```

A component is a physical object which has only one material but may have more than one manufacturing process. The material is not bound into the component: a material object contains no information which is specific to a particular component, so several different components may refer to the same material object. The manufacturing process, however, contains information which is uniquely identified with a particular component and is thus bound into the component.

The `piece_cost_fn()` method calculates the `piece_cost` by adding the material and processing costs for all the manufacturing processes. The `tool_cost_fn()` method calculates the `tool_cost` by adding the tool costs for all the manufacturing processes. If the `manufacturing_process` link is UNKNOWN or NONE then both `Component::piece_cost_fn()` and `Component::tool_cost_fn()` return values of UNKNOWN.

D.8.4 MechanicalPart

```
class MechanicalPart isa Component
 attribute variable geometry : bound Geometry
 attribute variable weight : FLOAT

 method weight_fn : FLOAT
 method piece_cost_fn : FLOAT
endclass

weight_fn() uses:
geometry.cubic_volume
material.specific_gravity


piece_cost_fn() uses:
manufacturing_process.material_cost
manufacturing_process.processing_cost
weight
material.cost_per_kg
```

Mechanical parts are components which also have a geometry associated with them. An example of a non-mechanical part is a user guide or manual for a product. The `weight_fn()` method calculates the weight by multiplying the `cubic_volume` of the geometry by the specific gravity of the material (a link to Material, `material`, is inherited from `Component`). The weight is stored in units of g, the cubic volume in units of cm^3 and the specific gravity in units of (g/cm^3). The `piece_cost_fn()` method calculates the piece cost by adding the result returned from `Component::piece_cost_fn()` to `(weight * material.cost_per_kg) / 1000`.

D.8.5 Assembly

```
class Assembly isa PhysicalObject
 attribute constant components : unbound PhysICO
 attribute variable assembly_process : bound AssemblyProcess[]

 method piece_cost_fn : FLOAT
 method tool_cost_fn : FLOAT
endclass

piece_cost_fn() uses:
components.piece_cost
(assembly_process).process_cost
(assembly_process).material_cost

tool_cost_fn() uses:
components.tool_cost
(assembly_process).tool_cost
```

User-defined classes derived from `Assembly` will have attributes and methods providing heuristic methods for estimating their piece, tooling and logistics costs in the absence (yet) of a breakdown into parts. The derivation routes for the inherited attributes piece, tooling and logistics costs will generally be as follows: highest priority (preferred) is adding up from components. Second priority (if components is UNKNOWN) is to use the heuristic rules defined in derived classes. If a point value is given for any attribute of any object then it always has the highest priority. If an `UncertainValue` is given, then its priority is given by its position in the list of derivation routes.

D.9 Classes related to PhysicalObject

D.9.1 Geometry

```
class Geometry isa SimWithID
 attribute variable cubic_volume : FLOAT
endclass
```

The Geometry class is used to store information regarding the geometry of a mechanical part. The cubic volume (used to calculate the weight of a mechanical part and its material cost) is stored in units of cm^3.

D.9.2 Material

```
class Material isa SimWithID
 attribute constant specific_gravity : FLOAT
 attribute constant cost_per_kg : FLOAT
endclass
```

Instances of material are not bound into components or assemblies since they do not contain any information which is specific to a particular component. Generally, a single instance exists in the model of each material available and it may be accessed by many different assemblies or components. The `specific_gravity` is stored in units of g/cm^3.

D.9.3 ManufacturingProcess

```
class ManufacturingProcess isa SimWithID
 attribute constant part_ref : unbound PhysicalObject
 attribute variable material_cost : FLOAT
 attribute variable process_cost : FLOAT
 attribute variable tool_cost : FLOAT

 method process_cost_fn : FLOAT
 method material_cost_fn : FLOAT
 method tool_cost_fn : FLOAT
endclass
```

Any physical object may have one or more manufacturing process associated with it. The tool cost of a manufacturing process (`ManufacturingProcess::tool_cost`) for a physical object represents the total cost of providing tools to perform the manufacturing process on the required number of objects. Suppose a design includes four identical panels each manufactured by pressing then painting. This would be modelled as a single instance of `Panel` with a cardinality of 4 in its `ICO` relationship. If it is decided that two spraying stations are required to paint the panels then `panel.painting.tool_cost` represents the total cost of providing two spraying stations. Tool costs are not multiplied by cardinalities in `PhysICO` relationships.

If `ManufacturingProcess` is instantiated directly, the material and processing costs (`ManufacturingProcess::material_cost` & `ManufacturingProcess::process_cost`) are entered directly by the user. The `process_cost_fn()`, `material_cost_fn()` and `tool_cost_fn()` methods must be implemented in derived classes - the versions supplied with `ManufacturingProcess` return values of UNKNOWN.

D.9.4 AssemblyProcess

```
class AssemblyProcess isa ManufacturingProcess
 attribute constant part_ref : unbound Assembly
 attribute variable material : unbound Material[]
 attribute variable matl_vol : FLOAT[]

 method material_cost_fn : FLOAT
endclass

material_cost_fn() uses:
       material.cost_per_kg
       material.specific_gravity
       matl_vol
```

Any `Assembly` may have one or more `AssemblyProcesses` associated with it. The class of `part_ref` is specialised to be an `Assembly` rather than a `PhysicalObject` - thus it is guaranteed to have a components attribute. A list of `AssemblyProcesses` is bound into `Assembly`. The `PhysICO` will aggregate the tool, piece and logistics costs for the components without using the `AssemblyProcess`, by querying the components themselves. But the assembly process will contribute its own tool, process and material costs which the `PhysICO` will also include in its aggregation. The `matl_vol` is the quantity of each material required for the assembly process, in units of cm^3. This class can be used as without further specialisation - the user is free to enter values for process and tooling costs directly.

The `material_cost_fn()` method calculates the material cost by multiplying `matl_vol` by the specific gravity and cost per kg of the material (and dividing by 1000.0 since specific gravity is stored in units of grams per cm^3). This value is aggregated over all materials consumed by the assembly process.

D.10 Relationship and derived classes

D.10.1 Relationship

```
class Relationship isa Sim
endclass
```

This is an abstract class which provides no methods, links or attributes and is used to represent relationships between other Sim objects. It should not be directly instantiated - only instances of its derived classes should be generated.

D.10.2 ICO and PhysICO

The `ICO` (is-a-component-of) classes are used to represent relationships between assemblies and their components.

```
class ICO isa Relationship
 attribute constant assembly : unbound SimWithID
 attribute variable components : bound SimWithID[]
 attribute variable cardinalities : INTEGER[]
 attribute variable unbound_components : unbound SimWithID[]
 attribute variable unbound_cardinalities : INTEGER[]
endclass

class PhysICO isa ICO
 attribute constant assembly : unbound Assembly
 attribute variable components : bound PhysicalObject[]
 attribute variable unbound_components : unbound PhysicalObject[]
 attribute constant piece_cost : FLOAT
 attribute constant logistics_cost : FLOAT
 attribute constant tool_cost : FLOAT

 method piece_cost_fn : FLOAT
 method logistics_cost_fn : FLOAT
 method tool_cost_fn : FLOAT
endclass


piece_cost_fn() uses:
        components).piece_cost
       cardinalities
       (unbound_components).piece_cost
       unbound_cardinalities

logistics_cost_fn() uses:
       (components).logistics_cost
       cardinalities
       (unbound_components).logistics_cost
       unbound_cardinalities

tool_cost_fn() uses:
       (components).tool_cost
```

The `PhysICO` class relates physical objects - but a model may contain other design entities (such as software or management task for example) which are not physical and require their own relationship classes to propagated their properties appropriately. These other relationships will be derived from `ICO`. There are two kinds of component in an `ICO`- bound and unbound. An object may only be a bound component of a single assembly, but it may appear as an `unbound_component` in many assemblies. The attributes `assembly`, `components` and `unbound_components` are specialised in `PhysICO` from being generic `SimWithIDs` to being an `Assembly` and `PhysicalObjects` respectively. Note that, since the components of `PhysICO` are `PhysicalObjects`, the `PhysICO` can aggregate sub-assemblies into assemblies as well as components into assemblies.

The `piece_cost_fn()` method calculates the total piece cost of the components (`PhysICO::piece_cost`) by adding the piece cost of all the components multiplied by their cardinality. Both bound and unbound component piece costs are aggregated. The method still returns a value if the value of either `components` or `unbound_components` (or indeed both) is NONE.

The `logistics_cost_fn()` method calculates the total logistics cost (`PhysICO::logistics_cost`) by adding the logistics cost of all the components multiplied by their

cardinality. Similarly to the `piece_cost_fn()` method, this method aggregates both the bound and the unbound component costs and still returns a value if either link has a value of NONE.

The `tool cost_fn()` method calculates the total tooling cost (`PhysICO::tool_cost`) by adding the tool cost of all the bound components, but their cardinality is ignored. The tooling cost of any unbound components is also ignored. If the value of components is NONE, this method returns a value of 0. This represents the situation where an assembly is composed entirely of components used elsewhere in the model, and thus incurs no additional tooling cost for the components.

D.10.3 IVO and PhysIVO

```
class  IVO isa Relationship
 attribute variable variants : unbound SimWithID[]
 attribute variable selected_variant : unbound SimWithID
endclass


class  PhysIVO isa IVO
  method piece_cost_fn ( ) : FLOAT
  method tool_cost_fn ( ) : FLOAT
  method logistics_cost_fn ( ) : FLOAT

 attribute variable variants : unbound PhysicalObject[]
 attribute variable selected_variant : unbound PhysicalObject
endclass


piece_cost_fn() uses:
        selected_variant).piece_cost

tool_cost_fn() uses:
        variants).piece_cost

logistics_cost_fn() uses:
       (selected_variant).piece_cost
```

This class is only required for models which include variants.

The `PhysIVO` class relates a set of physical objects which are variants - but a risk model may contain other design entities (software for example) which are not physical and require their own relationship classes. The `selected_variant` links in all the `IVOs` in a risk model are automatically assigned values by RiTo.

The `piece_cost_fn()` method calculates the piece cost over the set of variant objects. This is simply the cost of the selected variant. If the selected variant is NONE then there is no cost - returns a value of 0. The `logistics_cost_fn()` method is precisely analogous.

The `tool_cost_fn()` method calculates the tool cost over the set of variant objects. This is the sum of the tool costs for each variant. This method will still return a value if one or more of the variants has a value of NONE.

D.11 Classes involved in variant modelling

All the classes in this section are only required for risk models which include variants.

D.11.1 ProductModel

```
class ProductModel isa EdWithID
 attribute variable feature_model : bound FeatureModel
 attribute variable component_model : bound ComponentModel
endclass
```

Risk models which contain variants always have a `ProductModel` as their top object. A risk model only contains one `ProductModel`.

D.11.2 FeatureModel

Risk models which contain variants always include a single instance of `FeatureModel`. The feature model defines all the products which will eventually be offered to the customer, and also the options available to the customer for each product. It also specifies which variants are required for which products -

thus by editing the feature model, the designer can ensure that the component model will be correctly "pruned" when a particular product is selected.

```
class FeatureModel isa EdWithID
 attribute variable products : bound Product[]
 attribute variable selected_product : unbound Product
endclass
```

D.11.3 ComponentModel

```
class ComponentModel isa EdWithID
 attribute variable artefact : bound SimWithID
endclass
```

Risk models which contain variants always include a single instance of `ComponentModel`. The component model defines the assemblies and components which make up the designed artefact, and the relationships between them.

D.11.4 Product

```
class  Product isa SimWithID
 attribute variable features : unbound Feature[]
 attribute variable basic_model : unbound Feature
 attribute variable valid_options : bound CustomerOption[]
 attribute variable sales_volume : INTEGER
endclass
```

A product is a particular configuration of the designed artefact, which a customer may purchase. A product provides a set of features. One feature which is "special" is the basic model. This differs from the other features, in that parts of the component model which are "switched-in" by the basic model may subsequently be "switched out" by other features in the product (and vice versa). This does not imply any inconsistency in the model. If a part of the component model which is switched in by an ordinary features is also switched out by another ordinary feature, then this does imply an inconsistency in the product definition and RiTo will be unable to assign it to be the selected product.

The valid options are options which will be available to the customer with this particular product - the choice of whether or not to include a particular option will be made at the time of purchase.

The sales volume is the estimated total quantity of the product which will ultimately be sold, with any combination of valid customer options.

D.11.5 Feature

```
class Feature isa EdWithID
 attribute variable affects : unbound IVO[]
 attribute variable selects : INTEGER[]
 attribute variable implies : unbound Feature[]
endclass
```

A feature describes the capability of a product in terms of function; it is part of the `FeatureModel`. A feature can make selections from `IVO` (is-a-variant-of) relationships in the `ComponentModel`. This is the mechanism whereby the choice of products (and hence features) can "prune" the component model to the desired configuration. The link called `affects` points to all the `IVOs` which are affected by the inclusion/exclusion of this particular feature. The attribute called `selects` indicates which variant object is selected from each of these `IVOs` - if one of the variants in the `IVO` is NONE, then it is perfectly legal for a `Feature` to select a value of NONE. The INTEGER value of selects is an index into the list of variants stored in the `IVO`.

The implied features are a set of other features which will always be included if this feature is included.

D.11.6 CustomerOption

Customer options are part of the `FeatureModel` and are bound into a single particular `Product`. The take-up percentage represents the percentage of the total sales volume for the product which (it is estimated) will include this option. If `include` is set TRUE, then this customer option will be included when RiTo

evaluates the `ComponentModel`. The link called `features` points to the collection of `Features` which this customer option provides.

```
class CustomerOption isa SimWithID
 attribute variable take_up_percentage : FLOAT
 attribute variable include  : BOOL
 attribute variable features : unbound Feature[]
endclass
```

D.11.7 VolCoeffs

```
class VolCoeffs isa Editable
 attribute variable per_product : INTEGER[]
 attribute variable affected_by_option : unbound CustomerOption[]
 attribute variable extra_per_option : INTEGEP[]
endclass
```

This class is used to store volume coefficients for objects in the component model. All the necessary instances of this class are automatically created and populated with data by RiTo.

Appendix E: RiTo's Data Formats

Contents

E.1 Schema File

This section describes the format of the schema file, SCHEMA.TXT. The schema file contains the class definitions for the risk model. The classes are defined using a syntax which is based on Fusion syntax. A class definition begins:

```
class <class > isa <parent class>
```

where <class> inherits from <parent class>, and ends:

```
endclass
```

Base-typed attributes are defined thus:

```
attribute <mutability> <att name> : <type> [ <cardinality> ]
```

where:

```
<mutability> = constant or variable
<type> = FLOAT or INTEGER or BOOL or Str
```

The <cardinality> field may contain any integer value or it may be omitted. If the square brackets are included but the <cardinality> field is omitted then the attribute has variable cardinality - the cardinality may differ from one instance of the class to the next and may be changed at run-time by the designer. If the square brackets are also omitted then the attribute has a fixed cardinality of 1. If the mutability is constant, this means that the value cannot be changed after the object has been created, whereas the designer may edit the value of an attribute with variable mutability.

Object-valued attributes (or *links*) are defined thus:

```
attribute <mutability> <att name> : <binding> <class> [ cardinality ]
```

where:

```
<binding> = bound or unbound
```

The <cardinality> has the same meaning as for a base-typed attribute. A bound object will be destroyed when the parent is destroyed and created when the parent is created. A new copy of the bound object is created in a copy of the parent. A link <link> of class <link class> belonging to class <owner class> may have its class redefined to <new link class> in a class derived from <class name>, provided that <new link class> is derived from <link class>. The binding and mutability must remain unchanged. Methods are defined thus:

```
method <method name> : <return type>
```

The schema file should not generally be edited by the designer. The addition of new classes, or of new methods to existing classes requires the re-compilation of the RITO DLL. The only change which may be made to the schema file by the designer at run-time is the addition of new attributes or links to existing classes. However, these new attributes and links cannot be used directly by any methods. If a new attribute or link is added to the schema, then an existing risk model can be updated as follows. Run RiTo, load the existing risk model into memory and then save it with a new name. The newly saved model will have the new attribute or link present with a value of UNKNOWN.

E.2 Object Repository File

The object repository data file has file extension .SIM and contains the risk model. This section describes its format and explains how to edit the risk model by editing the object repository file. When you have finished editing a object repository file, it is advisable to open it in RiTo and then save it with a different name. The new file, saved by RiTo, will have the objects, attributes and links ordered optimally for speed when loading. Also, any attributes or links which were omitted from the original file will have UNKNOWN entries and this will also speed file loading.

E.2.1 Object Entries

There is an entry in the object repository file for every object in the risk model. The order of the object entries within the file affects the speed with which the object repository file is loaded into memory but is otherwise unimportant. An example of a complete object entry is shown below. Every object in the model has an *object id.* This is a unique identifier (unique within the object repository file) for the object and is used to define links to it. The object ID must be given in the first line of the object entry, appended to the string "`Object :`" . The object id is 0 in the example shown below. The second line of the object entry must contain the class of the object, appended to the string "`Class :`" . The class name is CegProduct in the example shown below:

```
Object  : 0
Class   : CegProduct
Str name = RoughingMill
Str description = The process area
INTEGER identifier = 0
FLOAT piece_cost = #att components.piece_cost
FLOAT tool_cost = #att components.tool_cost
FLOAT logistics_cost = #att components.logistics_cost
FLOAT sw_cost = #att components.sw_cost
FLOAT man_cost = #att components.man_cost
FLOAT integration_cost = #att components.total_int_cost
FLOAT total_cost = #method total_cost_fn
CegICO components = 2
```

Following the first two lines, there may be an entry for each attribute and each link of the object. If there is an attribute, given by the schema, but with no entry in the object entry, then RiTo will assign a value of UNKNOWN to that attribute. Similarly, if a link entry is omitted from the object entry then RiTo will assign a value of UNKNOWN to the link.

The order of the attribute and link entries within the object entry affects the speed with which the object repository file is loaded into memory but is otherwise unimportant. The end of an object entry is signalled by the beginning of the next object (a line beginning "`Object :`") or by the end of the file. All object repository files must contain an object with id 0. This is the *top object.*

E.2.2 Link entries

A link entry begins with the class of the link as given in the schema, followed by its name and the string "=". The object id of the object pointed to is then shown. If the link has cardinality greater than 1, then the object ids are shown separated by the string ", ". If the link is UNKNOWN then the string "**`Unknown`**" is shown and if the link is known to be NONE then the string "**`None`**" is shown. In the example below, the link `components_ico` has a cardinality of five. The objects with id 2, 3, 101, 5 and 6 must all be defined in the object repository file and must all either be of class `PhysicalObject` or a derived class or be IAO (is-alternative-of) objects. See Section E.2.4 for details of IAO links.

```
PhysicalObject components_ico = 2, 3, 101, 5, 6
```

To summarise, the syntax for a link entry is described below using the Fusion lifecycle syntax (an extended BNF notation, described in Chapter 6):

LinkEntry := *LinkClass LinkName* " = " (*LinkRoute* ", ") * *LinkRoute*

LinkRoute := "**None**" | "**Unknown**" | *ObjectID*

E.2.3 Attribute entries

E.2.3.1 Attributes with cardinality of 1

An attribute entry begins with the type of the attribute (FLOAT / INTEGER / BOOL / Str) followed by its name and the string " = ". The derivation routes for the attribute are then shown. They are shown in order

of priority with the most preferred route first, separated by the string " | ". The exception to this rule is point values - a point value is always interpreted by RiTo as the most preferred route, regardless of the position it occupies in the attribute entry. A derivation route may be a method, another attribute, a point value or an uncertain value.

A point value is simply written in ASCII format. For floating point values, the value may be written in full with a decimal point if needed or exponential format may be used if preferred. For example, any of the following are acceptable:

```
FLOAT my_attribute = 1000.0
FLOAT my_attribute = 1000
FLOAT my_attribute = 1E3
FLOAT my_attribute = 1e3
FLOAT my_attribute = 1000.003
```

A BOOLEAN value should be represented by the string "TRUE" or the string "FALSE". String values are not enclosed in quotes. Integer values may not be expressed in exponential format.

A derivation route which is a method is written with the string "**#method** " followed by the links to be followed, separated by ".", followed by the name of the method. For example, to use the `total_cost_fn()` method on this object:

```
FLOAT total_cost = #method total_cost_fn
```

Or, to use the `total_cost_fn()` method on object A, the object which is pointed to by the link `my_link` of this object:

```
FLOAT total_cost = #method my_link.total_cost_fn
```

Or, to use the `total_cost_fn()` method on object B, the object which is pointed to by link `his_link` of A:

```
FLOAT total_cost = #method my_link.his_link.total_cost_fn
```

When following a chain of links in this way, the links must be defined as part of the schema, not just part of the run-time class. For example, if `my_link` has class `AClass` and object A is of class `CClass`, derived from `AClass` by inheritance, then only those links defined in `AClass` may be followed from this object, not any additional links defined in `CClass`. Similarly, only those methods provided by the link class may be used. However, if a method called `total_cost_fn` is provided in `AClass` and re-implemented in `CClass`, then the `CClass` version will be used by the line:

```
FLOAT total_cost = #method my_link.total_cost_fn
```

A derivation route which is another attribute is written with the string "**#att** " followed by the links to be followed, separated by ".", followed by the name of the attribute. For example, to use the attribute `tooling_cost` of the object pointed to by attribute `components` of this object:

```
FLOAT tooling_cost = #att components.tooling_cost
```

The rules for following links are the same as for methods, but attributes (unlike methods) cannot be re-implemented in derived classes.

A derivation route which is an `UncertainValue` is written with the string "**#uncertain** " followed by the object id of the `UncertainValue` object. For example, the bandwidth of the control software shown below is a triangular probability distribution with minimum value of 5, peak value of 10 and maximum value of 20.

```
Object   : 6
Class    : CegControlSW
Str name = Control Software
Str description = s/w for PLCs and workstations
INTEGER identifier - 0
FLOAT sw cost    #att components.sw_cost
FLOAT bandwidth = #uncertain 120
'egSWICO components = 1 4

Object   : 120
Class    : TriangleFloat
FLOAT min = 5
FLOAT likely = 10
FL AT max - 20
```

If an attribute value is UNKNOWN then this should be represented by the string "**Unknown**".

The example below shows that the preferred route for the `piece_cost` is to use the value provided by the components. The second priority route (used if `components.piece_cost_ico` is UNKNOWN) is an uncertain value defined in the object with id 31.

```
 bject   : 4
Class    : Assembly
Str name - Trim
Str description = Unknown
INTEGER identifier = 0
FL AT piece_cost = #att components.piece_cost #uncertain 31
FLOAT tool_cost = #att components.tool_cost|#uncertain 34
FLOAT logistics_cost = #att components.logistics_cost #uncertain 35
SimWithID new_link = Unknown
PhysICO components = 12
AssemblyProcess assembly_process = 13
```

E.2.3.2 Attributes with cardinality greater than 1

If an attribute has cardinality greater than 1, the derivation routes for each value are shown separated by a ", ". For example, `probs`, below has a cardinality of three. The cardinality must be consistent with that given in the schema. If the schema defines a variable cardinality then any positive integer value (up to 65, 534) is acceptable. Otherwise, the cardinality must be the same as that given in the schema.

```
FLOAT probs = 0.2, 0.5, 0.3
```

E.2.3.3 Summary of attribute derivation route format

To summarise, the syntax of a single derivation route string for an attribute is described below using the Fusion lifecycle notation:

Route ::= "**#method** " (*Link* ".") * *MethodId* |
"**#att** ". (*Link* ".") * *AttName* ["[" [*AttCard*] "]"] |
"**#uncertain** " *ObjectID*
PointValue |
"**Unknown**"

MethodId ::= [*ClassName* ":"] *MethodName*

Link ::= *LinkName* ["[" [*LinkCard*] "]"]

AttName and *LinkName* are names of attributes and links respectively as given by the schema, and *AttCard* and *LinkCard* are cardinalities. These names and cardinalities must be consistent with the schema - RiTo checks that they are consistent and will not load the data file if they are not. A method may be identified simply by its name *MethodName* or by a class *ClassName* in addition to the method name. If no *ClassName* is given then the implementation of the method which is used in evaluating the route will depend upon the run-time class of the object - the implementation used will be that provided by the lowest class in the inheritance hierarchy to which the object belongs. If a *ClassName* is specified in the link then the method provided by that class will be used - even if the method has been "overridden" in a derived class to which the object belongs.

In Fusion lifecyle notation, the syntax for a complete attribute entry is:

AttributeEntry := *AttributeType AttName* " = " (*RouteList* ",") * *RouteList*

AttributeType := **"INTEGER" | "BOOL" | "FLOAT" | "Str"**

RouteList :- (*Route* " | ") * *Route*

E.2.4 Links to "Is-Alternative-Of" objects

Alternative designs, only one of which will eventually be chosen, are modelled using the `IAO` class and its derived classes. Any link in the object repository file may point to an object of class derived from `IAO`. An example of an `IAO` object of class `RandomIAO` is shown below:

```
bject  : 99
Class   : RandomIAO<Material>
Editable alternatives - 10, 11
Editable selection = #method choose_random
Discrete probabilities = 100
```

The class name has been augmented by the addition of the *IAO parameter class*, in this case `Material`, enclosed in triangular brackets. This is the minimum class of `Sim` object related by an `IAO` object. An `IAO` object may relate objects whose class is the same as, or derived from, the `IAO`'s parameter class. Any link of class `Material` (or a class above `Material` in the inheritance hierarchy) may point to this object. And objects 10 and 11 must be of class `Material` or a derived class.

The only situation where a derivation route for a link may be a method is the link `selection` in an `IAO` object. This is also the only situation where a method returns an object rather than a base-typed value. An example of a method for selecting alternatives is the `choose_random()` method, which belongs to class `RandomIAO` and which selects at random according to the probabilities stored in the `UncertainValue` object pointed to by `probabilities`. The `selection` link may be assigned a point value if you wish to evaluate the model for one of the alternatives.

Appendix F: Single Point Experiments Yield Misleading Results

What is meant by a single point experiment is one where only a single value is used for one or more variables during the risk sensitivity analysis. Two such algorithms are illustrated here on a very simple example. Consider the case where a risk model consists simply of three alternative assemblies A1, A2 and A3 with probabilities of 33%, 34% and 33% respectively and each assembly has an associated cost. Suppose that the cost of A1 is a point value of £5, the cost of A3 is £6 but the cost of A2 is only known to lie somewhere between £2 and £10 and is thus represented by a uniform distribution.

Pr(A1) = 33% A1.cost = 5
Pr(A2) = 34% A2.cost = UniformFloat(2, 10)
Pr(A3) = 33% A3.cost = 6

If *s1* is the risk sensitivity of the total cost to the IAO rv and *s2* is the risk sensitivity of the total cost to the cost of A2, then fixing one variable at a time at its expected value gives the following values for the risk sensitivities:

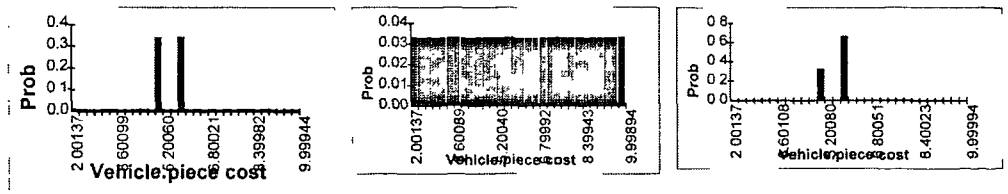


Figure 1: Output from simulations #1, #2 and #3 when A2 has highest probability

Simulation #1, Initial simulation : σ^2(total cost) = 2.00
Simulation # 2, IAO choice fixed at A2 : σ^2(total cost) = 5.33, *s1* = 2.00 - 5.33 = -3.33
Simulation # 3, cost of A2 fixed at £6 : σ^2(total cost) = 0.22, *s2* = 2.00 - 0.22 = 1.78

These results indicate strongly that the major source of uncertainty is the uncertainty in the cost of A2. The first problem withthe algorithm which this simple example illustrates is that it can yield a negative risk sensitivity value, and it is not at all clear how this should be interpreted. Now consider the effect of changing the probabilities by just 1%, so that A1 becomes marginally more probable than A2.

Pr(A1) = **34%** A1.cost = 5
Pr(A2) = **33%** A2.cost = UniformFoat(2, 10)
Pr(A3) = 33% A3.cost = 6

This time, we obtain the following values for the risk sensitivities. The distributions for total cost are as in Figure 1, except that for Simulation #2, the result is now a point value of 5.

Simulation #1, Initial simulation : σ^2(total cost) = 1.96
Simulation # 2, **IAO choice fixed at A1** : *s1* = 1.96 - 0 = 1.96
Simulation # 3, cost of A2 fixed at £6 : *s2* = 1.96 - 0.22 = 1.74

These results indicate weakly that the major source of uncertainty is the uncertainty in the choice between A1, A2 and A3. Thus it can be seen that a tiny change of 1% has reversed the ranking of the inputs generated by the algorithm. The fundamental problem here is that the choice of fixed value for the IAO is almost arbitrary. Next consider the results obtained if an alternative algorithm were used; where the inputs are varied one at a time, rather than being fixed one at a time. All other inputs except the one being varied are held at their expected values. The only difference in this particular example (where there are only two random variable inputs) is that the results don't have to be subtracted from an initial overall variance of 2.00, and thus -ve results cannot be obtained. Results obtained are:

	most likely alt is A2	**most likely alt is A1**
s1 = risk sensitivity to IAO	0.22	0.22
s2 = risk sensitivity to A2.piece_cost	5.33	0

Once again, the problem is the arbitrary choice of fixed value. This is the reason that single point experiments can yield misleading results.

Appendix G: Cegelec Specific Classes

G.1 Introduction and notation

The inheritance hierarchy for the Cegelec-specific classes is shown in Fusion notation in Figure G-1, Figure G-2 and Figure G-3. For each Cegelec-specific class this Appendix contains a textual description, and a Fusion class definition. The class definitions are reproduced from the schema file - see Appendix E for an explanation of the class definition notation. The remainder of the notation used in this Appendix is the same as that given in Appendix D, where the base classes are described.

G.2 Inheritance diagrams

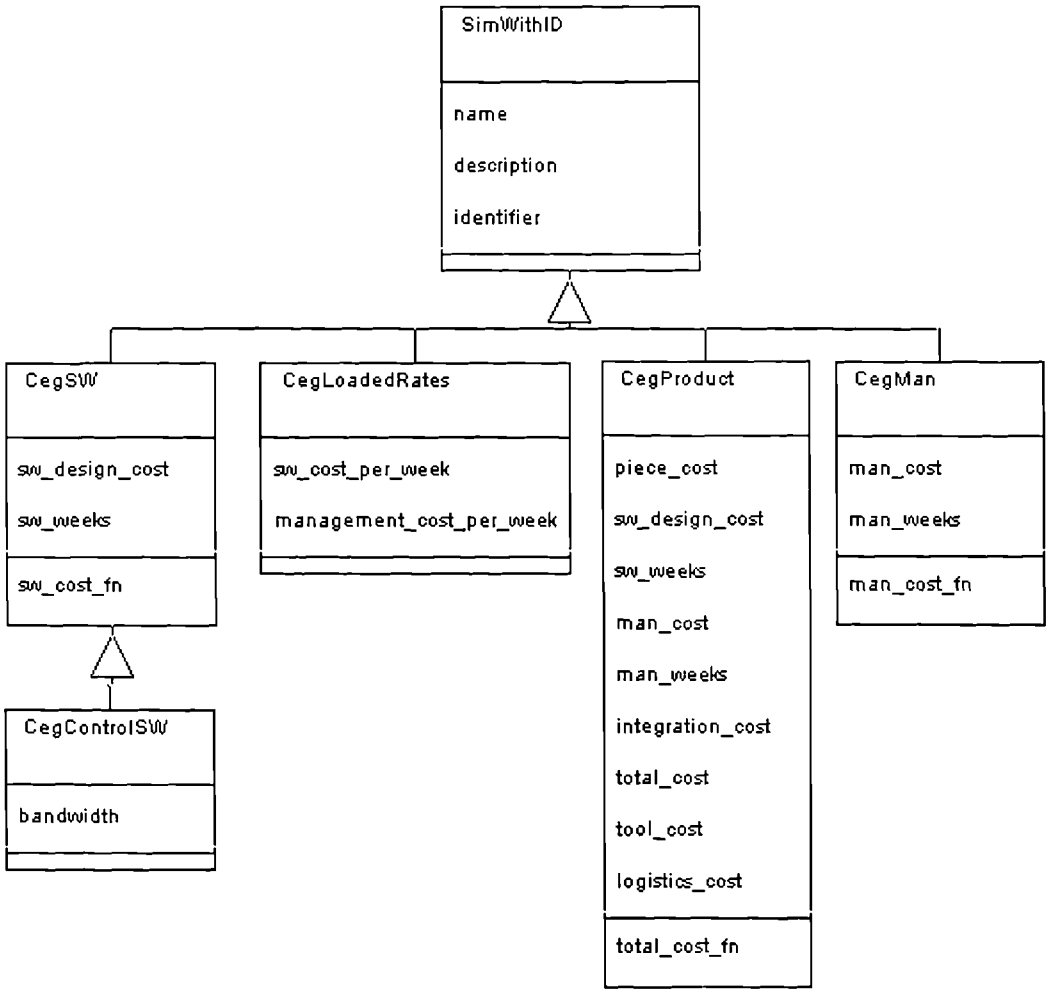


Figure G-1: Inheritance graph for SimWithID classes.

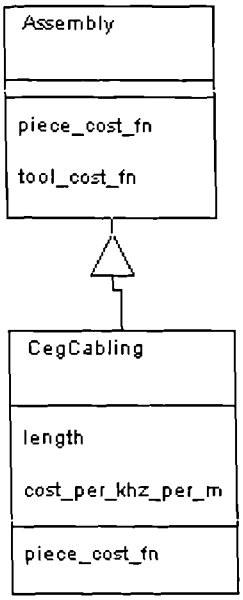


Figure G-2: Inheritance graph for hardware.

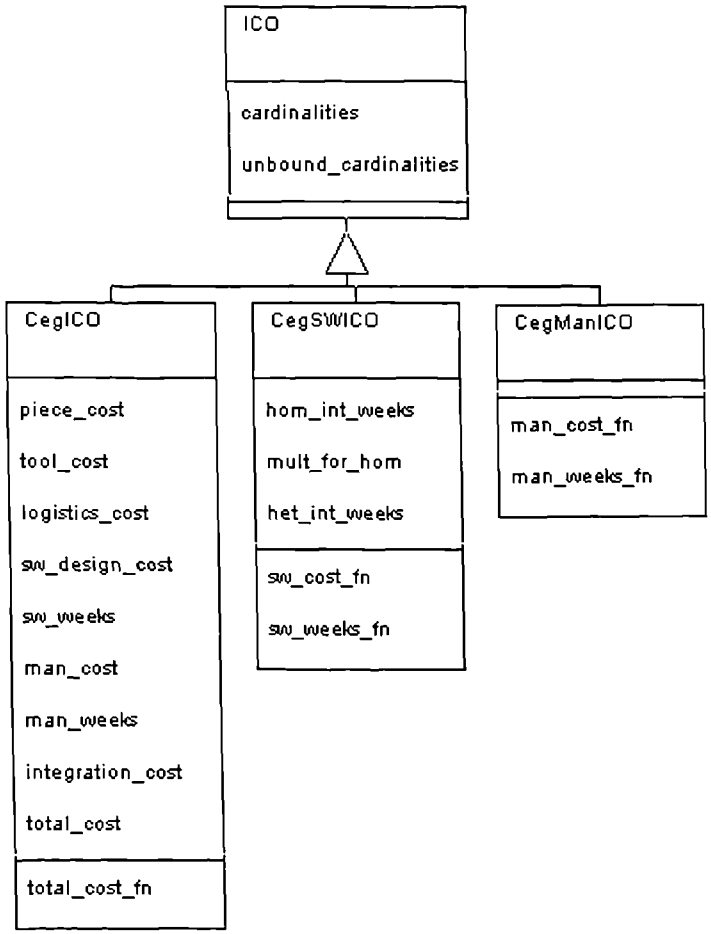


Figure G-3: Inheritance graph for ICO classes

G.3 Software Classes

G.3.1 CegSW

```
class CegSW isa SimWithID
 attribute variable sw_cost : FLOAT
 attribute variable sw_weeks : FLOAT
 attribute variable loaded_rates : unbound CegLoadedRates
 attribute constant components : unbound CegSWICO

 method sw_cost_fn : FLOAT
endclass

sw_cost_fn() uses:
        sw_weeks
        loaded_rates.sw_cost_per_week
```

An instance of `CegSW` represents an identifiable software entity which is required for the design - it could be an indivisible module, or it could be divisible into smaller software entities. The attribute `sw_weeks` represents the number of weeks of programming effort required to build the software entity. The attribute `sw_cost` represents the financial cost of carrying out this work. The link `loaded_rates` points to an object which stores the cost per unit time (per week) of programming effort. The link `components` is a link to an is-a-component-of relationship for software. It points to the software components via a CegSWICO, if the software entity is decomposed into smaller entities. If the software entity is indivisible, it takes a value of NONE and if the software entity hasn't yet been decomposed, it takes a value of UNKNOWN. The method `sw_cost_fn()` calculates `sw_cost` by multiplying `sw_weeks` by the cost per week of programming effort (which is stored in `loaded_rates`).

G.3.2 CegLoadedRates

```
class CegLoadedRates isa SimWithID
 attribute variable sw_cost_per_week : FLOAT
 attribute variable management_cost_per_week : FLOAT
endclass
```

The weekly costs stored in loaded rates represent the total cost to the design company of providing one person-week of work. Only a single instance of `CegLoadedRates` will generally exist in the model - although it may be useful to model alternatives if, for example, several outside contractors are under consideration and they each charge a different amount per week.

G.3.3 CegSWICO

```
class CegSWICO isa ICO
 attribute constant assembly : unbound CegSW
 attribute variable components : bound CegSW[]
 attribute variable hom_int_weeks : FLOAT[]
 attribute variable mult_for_hom : FLOAT[]
 attribute variable het_int_weeks : FLOAT[]

 method sw_cost_fn : FLOAT
 method sw_weeks_fn : FLOAT
endclass

sw_cost_fn() uses:
        assembly.loaded_rates.sw_cost_per_week
         components.sw_cost
         cardinalities
         het_int_weeks
         hom_int_weeks
         mult_for_hom

sw_weeks_fn() uses:
         components.sw_weeks
         cardinalities
         het_int_weeks
         hom_int_weeks
         mult_for_hom
```

A software is-a-component-of relationship relates a software entity to its components. The attributes "assembly" and "components" are both inherited from `ICO` but are specialised in the `CegSWICO` class definition. The cardinality of the components is also inherited from `ICO`.

For each component, values are stored for the homogeneous integration time, the heterogeneous integration time and a multiplier for the homogeneous integration time. The meaning of these three attributes is as follows: The homogeneous integration time is the number of weeks of effort required per component to integrate N identical components (where N is the cardinality of the component). The heterogeneous integration time is the number of weeks of effort required to integrate N of these components into the software assembly. The multiplier models economies of scale. It takes a value between 0 and 1 and is applied to the number of weeks required for each identical component after the first.

The `sw_cost_fn()` method calculates the software cost in currency units by aggregating the cost of the components and the integration costs as shown in pseudo-code below. The het and hom integration costs are given in units of weeks, and the loaded rate for the assembly (`assembly.loaded_rates`) is used to convert them to currency.

```
w = assembly.loaded_rates.sw_cost_per_week
FOR i = 0 TO number_of_components
{
 c = components[i].sw_cost
 result = result + het_int_weeks[i] * w  + c

 IF cardinality[i] > 1
  result = result +
   (cardinality[i] - 1) * (hom_int_weeks[i] * w + mult_for_hom[i] * c)
}
return result
```

G.3.4 CegControlSW

```
class CegControlSW isa CegSWAssy
 attribute variable bandwidth : FLOAT
endclass
```

The `CegControlSW` class provides an extra attribute "bandwidth" which is required to model the dependency of the optical cabling cost on the bandwidth required by the control sw.

G.4 Management Classes

G.4.1 CegMan

```
class CegMan isa SimWithID
 attribute variable man_cost : FLOAT
 attribute variable man_weeks : FLOAT
 attribute variable loaded_rates : unbound CegLoadedRates
 attribute constant components : unbound CegManICO

 method man_cost_fn : FLOAT
endclass

man_cost_fn() uses:
        man_weeks
        loaded_rates.man_cost_per_week
```

An instance of `CegMan` represents a management task which is required for the design - similarly to an instance of `CegSW`, it may either be an atomic unit or it may be divisible into smaller sub-tasks. The attribute `man_weeks` represents the number of weeks of management effort required to complete the task. The attribute `man_cost` represents the financial cost of carrying out the work. The link `loaded_rates` points to an object which stores the cost per unit time (per week) of management effort. The link `components` is a link to an is-a-component-of relationship for mangement tasks. It points to the component tasks via a CegManICO if the task is decomposed into sub-tasks. If the task is indivisible it takes a value of NONE and if it has not yet been decomposed it takes a value of UNKNOWN. The method `man_cost_fn()` calculates `man_cost` by multiplying `man_weeks` by the cost per week of management effort (which is stored in `loaded_rates`).

G.4.2 CegManICO

```
class CegManICO isa ICO
 attribute constant assembly : unbound CegMan
 attribute variable components : bound CegMan[]

 method man_cost_fn : FLOAT
 method man_weeks_fn : FLOAT
endclass

man_cost_fn() uses:
        components.man_cost
        cardinalities

man_weeks_fn() uses:
        components.man_weeks
        cardinalities
```

A management is-a-component-of relationship relates a management task to its component tasks. The attributes "assembly" and "components" are both inherited from `ICO` but are specialised in the `CegManICO` definition. The cardinality of the components is also inherited from `ICO`.

The method `man_cost_fn()` calculates the total management cost for the components by aggregating the cost of the components multiplied by their cardinality. No integration costs are modelled for management tasks.

The method `man_weeks_fn()` calculates the total number of weeks required for the component management tasks by aggregating the number of weeks for the components multiplied by their cardinality.

G.5 Design Module Classes

G.5.1 CegProduct

```
class CegProduct isa SimWithID
 attribute constant components : unbound CegICO
 attribute variable piece_cost : FLOAT
 attribute variable tool_cost : FLOAT
 attribute variable logistics_cost : FLOAT
 attribute variable sw_cost : FLOAT
 attribute variable sw_weeks : FLOAT
 attribute variable man_cost : FLOAT
 attribute variable man_weeks : FLOAT
 attribute variable integration_cost : FLOAT
 attribute variable total_cost : FLOAT

 method total_cost_fn : FLOAT
endclass

total_cost_fn() uses:
        piece_cost
        tool_cost
        logistics_cost
        sw_cost
        man_cost
        integration_cost
```

A `CegProduct` (or "design module") may contain, via its components, software (`CegSW`), hardware (`PhysicalObject`), management tasks (`CegMan`) and one or more other design modules.

NOTE: `CegProduct` does not inherit from `PhysicalObject`, although this would have avoided the need to define piece, tooling and logistics costs in the `CegProduct` class definition, because it is not correct to say that a design module is "a kind of `PhysicalObject`".

The piece, tooling, logistics, management and software cost attributes have their usual meaning. The integration cost is the total cost of integrating hardware and software for this design module and any other design modules which it contains. The total cost is a local total of all six cost attributes, calculated by the `total_cost_fn()` method. The total cost is not propagated through the hierarchy. The management time attribute (`man_weeks`) represents the total amount of time required for all management tasks in this design module and any other design modules or management tasks which it contains. Similarly, `sw_weeks` represents the total number of weeks of programming effort required for this design module and any other design modules or software entities which it contains.

G.5.2 CegICO

```
class CegICO isa ICO
 attribute variable software : bound CegSW
 attribute variable hardware : bound PhysicalObject
 attribute variable management : bound CegMan
 attribute variable components : bound CegProduct[]
 attribute variable assembly : unbound CegProduct
 attribute variable integration_cost : FLOAT

 method total_int_cost_fn : FLOAT
 method piece_cost_fn : FLOAT
 method tool_cost_fn : FLOAT
 method logistics_cost_fn : FLOAT
 method sw_cost_fn : FLOAT
 method sw_weeks_fn : FLOAT
 method man_cost_fn : FLOAT
 method man_weeks_fn : FLOAT
endclass

total_int_cost_fn() uses:
        integration_cost
        (components).integration_cost

piece_cost_fn() uses:
        (hardware).piece_cost
        (components).piece_cost
        cardinalities

tool_cost_fn() uses:
        (hardware).tool_cost
        (components).tool_cost

logistics_cost_fn() uses:
        (components).logistics_cost
         hardware).logistics_cost

sw_cost_fn() uses:
        (software).sw_cost
        (components).sw_cost

sw_weeks_fn() uses:
        (software).sw_weeks
        (components).sw_weeks

man_cost_fn() uses:
        (management).man_cost
        (components).man_cost

man_weeks_fn() uses:
        (management).man_weeks
        (components).man_weeks
```

The `CegICO` is-a-component-of relationship relates a `CegProduct` assembly to its `CegProduct` components and also to its software, hardware and management components. The attributes `assembly` and `components` are both inherited from `ICO` but are specialised in the `CegICO` definition. The cardinality of the components is also inherited from `ICO`.

The attribute `integration_cost` represents the cost of integrating the hardware and software for this particular relationship. The `total_int_cost_fn()` method calculates the total integration cost by aggregating the integration costs of the `CegProduct` components. This is unaffected by the cardinality of the components - the cost of integrating hardware and software into a `CegProduct` is incurred once for a given `CegProduct` object instance, regardless of how many duplicates of the object are actually supplied. It is legal to have a `CegICO` with `components` equal to NONE, and in this case just the local integration cost is used.

The `piece_cost_fn()` method calculates the `piece_cost` by aggregating the piece costs of the `CegProduct` components and the hardware, multiplying by the cardinality of the components. If the hardware or components are equal to NONE, then they are omitted from the aggregation.

The `tooling_cost_fn()` method calculates the tooling cost by aggregating the tooling costs of the `CegProduct` components and the hardware, without multiplying by the cardinality of the components. If the hardware or components are equal to NONE, then they are omitted from the aggregation.

The `logistics_cost_fn()` method calculates the logistics cost by aggregating the logistics costs of the `CegProduct` components and the hardware, multiplying by the cardinality of the components. If the hardware or components are equal to NONE, then they are omitted from the aggregation.

The `sw cost_fn()` method calculates the sw_cost by aggregating the software costs of the `CegProduct` components and the software. The cardinality of the `CegProduct` components is ignored - it is assumed that the software cost associated with a `CegProduct` is incurred once, regardless of how many copies of the product are supplied.

The `sw_weeks_fn()` method calculates the total number of weeks required to build the software for the assembly and all of its sub-assemblies. It adds the software weeks from its `CegProduct` components (`components.sw_weeks`) and its software (`software.sw_weeks`). The cardinality of a `CegProduct` component is ignored - it is assumed that the time required to build software for a `CegProduct` is incurred once, regardless of how many copies of the product are supplied. Note that the value returned by this method may be independent of the value returned by `sw_cost_fn()`, since the components may not all use the same loaded rate.

The `man_cost_fn()` method calculates the management cost by aggregating the management costs of the `CegProduct` components and the hardware, ignoring the cardinality of the `CegProduct` components.

The `man_cost_fn()` method calculates the total number of weeks required for management tasks for the assembly and all of its sub-assemblies. It adds management weeks from its `CegProduct` components (`components.man_weeks`) and its management (`management.man_weeks`). The cardinality of a `CegProduct` component is ignored - it is assumed that the time required for management tasks for a `CegProduct` is incurred once, regardless of how many copies of the product are supplied.

G.6 PhysicalObject Classes

G.6.1 CegCabling

```
class CegCabling isa Assembly
 attribute variable length : FLOAT
 attribute variable cost_per_khz_per_m : FLOAT
 attribute variable control_sw : unbound CegControlSW
 method piece_cost_fn : FLOAT
endclass

piece_cost_fn() uses:
        control_sw.bandwidth
        length
        cost_per_khz_per_m
```

An instance of cabling represents the cabling required for part or all of a control installation. It is an assembly, and so may contain other physical objects and have an assembly process. The length of the cabling required and its cost per KHz of bandwidth per metre of length is stored. There is a link to the control software with which it will be used and the `piece_cost_fn()` method calculates the piece cost of the cabling using the bandwidth required by the control software.

Appendix H: Exemplar Rover Specific Classes

Contents

H.1 Introduction and notation

The inheritance hierarchy for the two exemplar Rover-specific classes is shown in Fusion notation in Figure H-1 and Figure H-2. For each Rover-specific class this Appendix contains a textual description, and a Fusion class definition. The class definitions are reproduced from the schema file - see Appendix E for an explanation of the class definition notation. The remainder of the notation used in this Appendix is the same as that given in Appendix D, where the base classes are described.

H.2 Inheritance diagrams

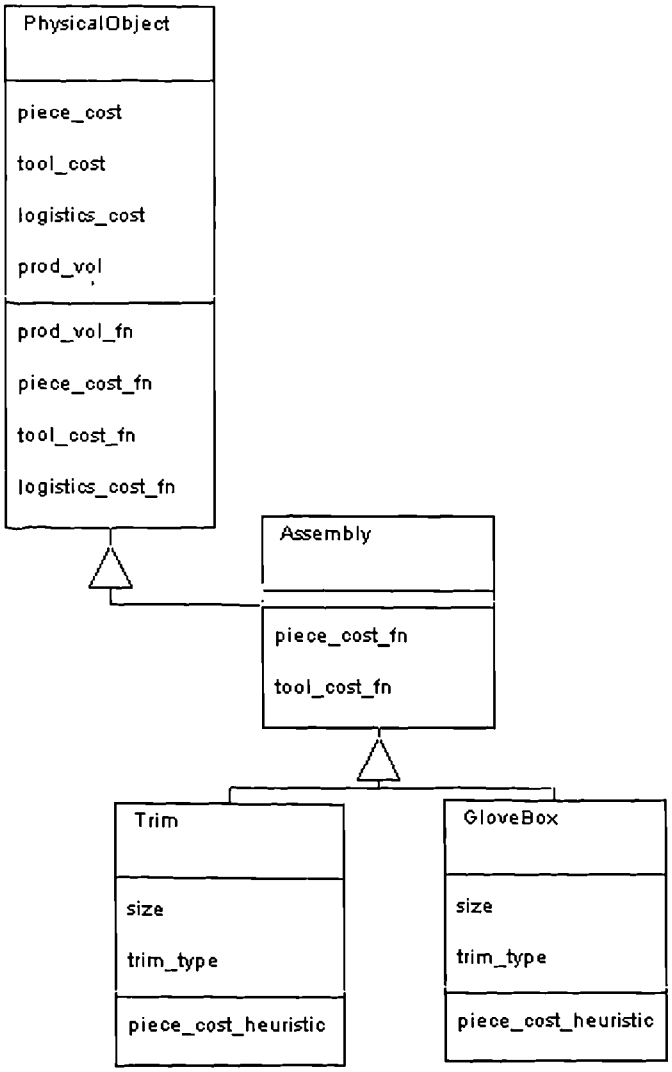


Figure H-1: Inheritance graph for exemplar classes.

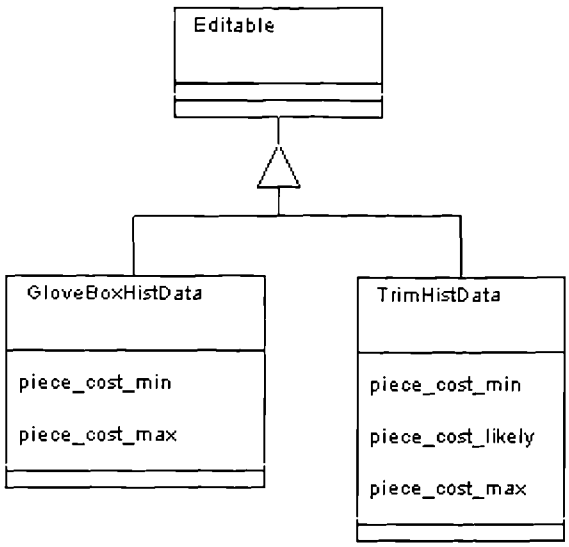


Figure H-2: Inheritance graph for historical data.

H.3 Class Definitions

H.3.1 GloveBox

```
class GloveBox isa Assembly
 attribute variable size : Str
 attribute variable trim_type : Str
 attribute variable hist_data : unbound GloveBoxHistData
 method piece_cost_heuristic : FLOAT
endclass

class GloveBoxHistData isa Editable
 attribute variable piece_cost_min : FLOAT[9]
 attribute variable piece_cost_max : FLOAT[9]
endclass
```

`piece_cost_heuristic()` uses:

```
size
trim_type
hist_data.piece_cost_min
hist_data.piece_cost_max
```

An instance of the `GloveBox` class is used to represent a glove box assembly in a vehicle. This class provides an example of an heuristic method. The method `Glovebox::piece_cost_heuristic()` takes as input strings representing the `size` and the `trim_type` and produces a uniform distribution (an instance of `UniformFloat`) as ouput. Strings are used here to represent an enumerated type (since RiTo does not yet support enumerated types):-

`size` {"large", "medium", "small"}
`trim_type` {"soft feel", "hard feel", "leather"}

The parameters of the distribution (min and max) which are selected for each combination of input values are stored in a instance of `GloveBoxHistData` (historical data regarding gloveboxes). Notice that this is an unbound link, because all instances of `GloveBox` will use the same historical data. The index into `piece_cost_min` and `piece_cost_max` for each combination of values is shown below:

	large	medium	small
soft feel	0	1	2
hard feel	3	4	5
leather	6	7	8

H.3.2 Trim

```
class Trim isa Assembly
 attribute variable car_size : Str
 attribute variable trim_type : Str
 attribute variable hist data : unbound TrimHistData
 method piece_cost_heuristic : FLOAT
endclass

class TrimHistData isa Editable
 attribute variable piece_cost_min : FLOAT[9]
 attribute variable piece_cost_likely : FLOAT[9]
 attribute variable piece_cost_max : FLOAT[9]
endclass
```

piece_cost_heuristic() uses:

```
car_size
trim_type
hist_data.piece_cost_min
hist_data.piece_cost_likely
hist_data.piece_cost_max
```

An instance of the `Trim` class is used to represent the trim area in a vehicle. This class provides an example of an heuristic method. The method `Trim::piece_cost_heuristic()` takes as input strings representing the `car_size` and the `trim_type` and produces a triangular distribution (an instance of `TriangleFloat`) as ouput. Strings are used here to represent an enumerated type (since RiTo does not yet support enumerated types):-

`car_size` {"large", "medium", "small"}
`trim_type` {"soft feel", "hard feel", "leather"}

The parameters of the distribution (min, likely and max) which are selected for each combination of input values are stored in a instance of `TrimHistData` (trim area historical data). Notice that this is an unbound link, because all instances of `Trim` will use the same historical data. The index into `piece_cost_min`, `piece_cost_likely` and `piece_cost_max` for each combination of values is shown below:

	large	medium	small
soft feel	0	1	2
hard feel	3	4	5
leather	6	7	8

Appendix I: Analytical Solution for HRSA Case 3

In this appendix, the risk sensitivity of $\boldsymbol{Y}$ = piece part cost to production volume is calculated for the third example (Case 3) given in Section 11.2.5 of Chapter 11. The risk sensitivity is defined as the expected value of the variance reduction in $\boldsymbol{Y}$, when the production volume is replaced by a fixed value. The expected value of the variance of $\boldsymbol{Y}$ for fixed production volume is calculated analytically and is then subtracted from the original variance of $\boldsymbol{Y}$. The original variance of $\boldsymbol{Y}$ is evaluated using Monte Carlo simulation.

Let the following be random variables:

$\boldsymbol{PV}$ = production volume
$\boldsymbol{C1}$ = cost of first component
$\boldsymbol{C2}$ = cost of second component
$\boldsymbol{P1}$ = cost of first assembly process
$\boldsymbol{P2}$ = cost of second assembly process

and let their respective probability distributions be:

$$F_{PV}(x) = \text{Triangle}_{2000,\ 12000,\ 40000}(x)$$

$$F_{C1}(x) = \text{Triangle}_{13.5,\ 17,\ 18.5}(x)$$

$$F_{C2}(x) = \text{Uniform}_{7.5,\ 9.25}(x)$$

$$F_{P1}(x) = \text{Triangle}_{7,\ 9,\ 10}(x)$$

$$F_{P2}(x) = \text{Uniform}_{5,11}(x)$$

And let the total piece cost, $\boldsymbol{Y}$, be given by:

Equation I-1

$$\boldsymbol{Y} = G_Y(\boldsymbol{PV}, \boldsymbol{C1}, \boldsymbol{C2}, \boldsymbol{P1}, \boldsymbol{P2}) = \begin{cases} \boldsymbol{C1} + \boldsymbol{C2} + \boldsymbol{P1} + 13.14, & \text{if } \boldsymbol{PV} \in [2000,\ 5000] = I_1 \\ \boldsymbol{C1} + \boldsymbol{C2} + \boldsymbol{P2} + 13.00, & \text{if } \boldsymbol{PV} \in (5000,\ 10000] = I_2 \\ \boldsymbol{C1} + \boldsymbol{C2} + \boldsymbol{P2} + 10.90, & \text{if } \boldsymbol{PV} \in (10000,\ 28000] = I_3 \\ \boldsymbol{C1} + \boldsymbol{C2} + 17.63, & \text{if } \boldsymbol{PV} \in (28000,\ 30000] = I_4 \\ \boldsymbol{C1} + \boldsymbol{C2} + 16.88, & \text{if } \boldsymbol{PV} \in (30000,\ 40000] = I_5 \end{cases}$$

And let:

$$V_Y(k) = \text{var}(G_Y \mid \boldsymbol{PV} = k)$$

It is necessary to calculate the risk sensitivity to production volume, *RS*, which is defined to be the original variance of $\boldsymbol{Y}$ minus expected value of this variance if $\boldsymbol{PV}$ is fixed. Calculating the expected value of the variance for fixed $\boldsymbol{PV}$:

$$\begin{aligned} \text{var}(\boldsymbol{Y}) - RS &= \text{E}\left[\text{var}(G_Y \mid \boldsymbol{PV} = k\right] \\ &= \int_{2000}^{4000} V_Y(k) \cdot F_{PV}(k)\,dk \end{aligned}$$

But $V_Y(k)$ is a step function (the variance only changes at certain discrete values of production volume) and thus:

Equation I-2

$$\text{var}(\boldsymbol{Y}) - RS = \sum_{j=1}^{5} \int_{I_j} V_{Y_j} \cdot \text{F}_{\text{PV}}(k)dk$$

$$= \sum_{j=1}^{5} V_{Y_j} \cdot \int_{I_j} \text{F}_{\text{PV}}(k)dk$$

where:

V_{Y1} = var(***C1*** + ***C2*** + ***P1***) = var(***C1***) + var(***C2***) + var(***P1***)
$V_{Y2} = V_{Y3}$ = var(***C1*** + ***C2*** + ***P2***) = var(***C1***) + var(***C2***) + var(***P2***)
$V_{Y4} = V_{Y5}$ = var(***C1*** + ***C2***) = var(***C1***) + var(***C2***)

Since variance is additive for independent random variables, the $\{V_{Yj}\}$ can be calculated as sums of the variances of the four triangular and uniformly distributed inputs. If x has a triangular distribution with min = a, likely = b, max = c, then:

$$\text{var}(\boldsymbol{x}) = \frac{a^2 + b^2 + c^2 - ab - ac - bc}{18}$$

And if x has a uniform distribution with min = a, max = b then:

$$\text{var}(\boldsymbol{x}) = \frac{(b-a)^2}{12}$$

Thus we obtain the following results:

random variable	variance
C1	1.097222
C2	0.255208
P1	0.388889
P2	3.000000

Table I-1: Variance of triangular and uniformly distributed inputs

and thus:

V_{Y1} = 1.741319
$V_{Y2} = V_{Y3}$ = 4.352430
$V_{Y4} = V_{Y5}$ = 1.352430

It is also necessary to calculate the integrals in Equation I-2, but each one is simply the area of the trapezoid which lies under the Triangular distribution function F_{PV} and is bounded by the interval Ij. Thus the following table can be calculated:

j	$\int_{I_j} \text{F}_{\text{PV}}(k)dk$
1	0.023684211
2	0.144736842
3	0.696240601
4	0.041353383
5	0.093984962

Table I-2: Probability of production volume falling within each interval

And therefore, using Equation I-2:

var(***Y***) - *RS* = 3.757466358

It now remains to calculate the original variance of Y. Looking at Equation I-1, it is clear that Y can be expressed as the sum of the three independent rvs:

$$Y = C1 + C2 + Y'$$

where Y' is given by:

$$Y' = \begin{cases} P1 + 13.14, & \text{if } PV \in [2000,\ 5000] = I_1 \\ P2 + 13.00, & \text{if } PV \in (5000,\ 10000] = I_2 \\ P2 + 10.90, & \text{if } PV \in (10000,\ 28000] = I_3 \\ 17.63, & \text{if } PV \in (28000,\ 30000] = I_4 \\ 16.88, & \text{if } PV \in (30000,\ 40000] = I_5 \end{cases}$$

It would be possible to calculate var(Y') analytically, but for simplicity it has been chosen instead to evaluate it directly by Monte Carlo simulation. Requiring that all statistics converge to within 0.5%, we obtain a value of approximately `4.09 +/- .02` for the variance of Y'. And thus:

$$\text{var}(Y) = \text{var}(C1) + \text{var}(C2) + \text{var}(Y') = 5.44 \text{ +/- } .02$$

and therefore the risk sensitivity to production volume is given by:

$$RS = 5.44 - 3.757466358 = 1.68 \text{ +/- } .02$$

Appendix J: Attribute Derivation Networks for Interior Trim Risk Models

This appendix contains the attribute derivation networks for the Interior Trim risk model examples presented in Chapter 11. Cases 2 - 5 are presented, which are used to illustrate the Hierarchical Risk Sensitivity Analysis. The networks were generated using the Audit Trail facility provided by RiTo (see Section 10.2.2.3 in Chapter 10).

J.1 Case 2: Piece cost of an Injection Moulding

```
INTEGRATEDARMATURE(14).piece_cost = #method piece_cost_fn
 INJECTION MOULDING(19).material_cost = #uncertain TriangleFloat(8.00,8.25,9.00)
 INJECTION MOULDING(19).process_cost = 20.30
 TRIMMING(20).material_cost = 0.00
 TRIMMING(20).process_cost = 0.50
 INTEGRATEDARMATURE(14).weight = #method weight_fn
  FACIA GEOMETRY(21).cubic_volume = #uncertain TriangleFloat(18.00,23.00,30.00)
  PLASTIC(18).specific_gravity = 12.50
 PLASTIC(18).cost_per_kg = #uncertain TriangleFloat(2.00,2.60,3.00)
```

J.2 Case 3: Choice of assembly process depends upon component production volume

```
NONINTEGRATEDARMATUREASSY(113).piece cost = #method piece cost fn
 PHYSICO(114).piece_cost = #method piece_cost_fn
  NONINTEGRATEDARMATURE(66).piece_cost = #uncertain TriangleFloat(13.500000,17.000000,18.500000)
  PHYSICO(114).cardinalities[0] = 1
  BINNACLEMOULDING(115).piece_cost - #uncertain UniformFloat(7.500000,9.250000)
  PHYSICO(114).cardinalities[1] - 1
 NONINTEGRATEDARMATUREASSY(113).prod vol = #uncertain TriangleFloat(2000.000000,12000.000000,40000.000000)
 BYVOLIAO(151).prod_vols[0] = 10000.000000
 BYVOLIAO(151).prod_vols[1] - 30000.000000
 ASSEMBLY PROCESS 1 LOW VOL(153).process_cost - 10.600000
 ASSEMBLY PROCESS 1 MED VOL(154).process_cost - 9.000000
 ASSEMBLY PROCESS 1 HIGH VOL(155).process_cost = 8.250000
 NONINTEGRATEDARMATUREASSY(113).prod_vol = #uncertain TriangleFloat(2000.000000,12000.000000,40000.000000)
 BYVOLIAO(152).prod_vols[0] = 5000.000000
 BYVOLIAO(152).prod_vols[1] = 28000.000000
 ASSEMBLY PROCESS 2 LOW VOL(156).process_cost = #uncertain TriangleFloat(7.000000,9.000000,10.000000)
 ASSEMBLY PROCESS 2 MED VOL(157).process_cost = #uncertain UniformFloat(5.000000,11.000000)
 ASSEMBLY PROCESS 2 HIGH VOL(158).process_cost = 6.800000
 NONINTEGRATEDARMATUREASSY(113).prod_vol = #uncertain TriangleFloat(2000.000000,12000.000000,40000.000000)
 BYVOLIAO(151).prod_vols[0]   10000.000000
 BYVOLIAO(151).prod_vols[1]   30000.000000
 ASSEMBLY PROCESS 1 LOW VOL(153).material cost - 1.500000
 ASSEMBLY PROCESS 1 MED VOL(154).material_cost   1.000000
 ASSEMBLY PROCESS 1 HIGH VOL(155).material_cost = 1.000000
 NONINTEGRATEDARMATUREASSY(113).prod_vol = #uncertain TriangleFloat(2000.000000,12000.000000,40000.000000)
 BYVOLIAO(152).prod_vols[0] = 5000.000000
 BYVOLIAO(152).prod_vols[1] - 28000.000000
 ASSEMBLY PROCESS 2 LOW VOL(156).material_cost - 1.040000
 ASSEMBLY PROCESS 2 MED VOL(157).material_cost = 0.900000
 ASSEMBLY PROCESS 2 HIGH VOL(158).material_cost = 0.825000
```

J.3 Case 4: Piece cost of an injection moulding with alternative manufacturing processes

```
INTEGRATEDARMATURE(14).piece_cost = #method piece_cost_fn
 DISCRETE(118).probs[0] = 0.333333
 DISCRETE(118).probs[1] = 0.333333
 DISCRETE(118).probs[2] = 0.333333
 INJECTION MOULDING 1(19).material_cost = 7.400000
 INJECTION MOULDING 2(128).material_cost = #uncertain TriangleFloat(7.200000,7.400000,8.280000)
 INJECTION MOULDING 3(1000).material_cost = 7.750000
 DISCRETE(118).probs[0] = 0.333333
 DISCRETE(118).probs[1] = 0.333333
 DISCRETE(118).probs[2] = 0.333333
 INJECTION MOULDING 1(19).process_cost = 20.300000
 INJECTION MOULDING 2(128).process_cost = 20.300000
 INJECTION MOULDING 3(1000).process_cost = 20.300000
 TRIMMING(20).material_cost = 0.000000
 TRIMMING(20).process_cost = 0.500000
 INTEGRATEDARMATURE(14).weight = #method weight_fn
  FACIA GEOMETRY(21).cubic_volume = #uncertain PiecewiseLinearFloat(0.000000,3.000000,0.000000,0.000000,1.000000,1.000000,
                                    0.000000,17.800000,18.000000,19.000000,25.000000,26.000000,27.000000,30.000000)
  PLASTIC(18).specific_gravity = #uncertain PiecewiseLinearFloat(0.000000,3.000000,1.000000,1.000000,0.000000,12.000000,
                                 12.100000, 12.400000,12.800000,13.000000)
 PLASTIC(18).cost_per kg - #uncertain PiecewiseLinearFloat(0.000000,3.000000,0.000000,0.000000,1.000000,1.0 0000,0.000000,
                           2.000000,2.100000,2.300000,2.650000,2.800000,2.900000,3.000000)
```

J.4 Case 5: Introduction of Variant Assemblies

```
INTERIORTRIM(18).piece_cost = #method piece_cost_fn
 PHYSICO(26).piece_cost = #method piece_cost_fn
  GENERIC FLOORCOVERINGS(83).piece_cost = #method variants.piece_cost fn
   LOW VOL PLAIN FLOORCOVERINGS(258).prod_vol = #method prod_vol_fn
    VOLCOEFFS(223).per_product[0] = 1
    PRODUCT A(13).sales_volume = #uncertain TriangleFloat(500.000000,5000.000000,10000.000000)
    VOLCOEFFS(223).per_product[1] = 0
    VOLCOEFFS(223).per_product[2] = 0
   BYVOLIAO(247).prod_vols[0] = 4000.000000
   BYVOLIAO(247).prod_vols[1] = 7000.000000
   LOW VOL PLAIN FLOORCOVERINGS(258).piece_cost = #uncertain TriangleFloat(47.000000,50.000000,55.000000)
   MED VOL PLAIN FLOORCOVERINGS(259).piece_cost = 35.000000
   HIGH VOL PLAIN FLOORCOVERINGS(260).piece_cost = 20.000000
  PHYSICO(26).cardinalities[0] = 1
  OVERALL ROOFTRIM(84).piece_cost = #method piece_cost_fn
   PHYSICO(296).piece_cost = #method piece_cost_fn
    LOW VOL ROOFTRIM(286).prod_vol = #method prod_vol_fn
     VOLCOEFFS(284).per_product[0] = 1
     PRODUCT A(13).sales volume = #uncertain TriangleFloat(500.000000,5000.000000,10000.000000)
     VOLCOEFFS(284).per_product[1] = 1
     PRODUCT B(1).sales_volume = #uncertain TriangleFloat(500.000000,20000.000000,35000.000000)
     VOLCOEFFS(284).per_product[2] = 1
     PRODUCT C(2).sales_volume = #uncertain TriangleFloat(500.000000,10000.000000,12000.000000)
    BYVOLIAO(295).prod_vols[0] = 25000.000000
    BYVOLIAO(295).prod_vols[1] = 35000.000000
    LOW VOL ROOFTRIM(286).piece cost - #uncertain TriangleFloat(33.000000,35.000000,36.500000)
    MED VOL ROOFTRIM(287).piece_cost = #uncertain TriangleFloat(36.000000,40.000000,44.000000)
    HIGH VOL ROOFTRIM(288).piece cost - #uncertain TriangleFloat(23.000000,25.000000,27.000000)
    PHYSICO(296).cardinalities = 1
  PHYSICO(26).cardinalities[1] = 1
  GENERIC SEATS(85).piece_cost = #method variants.piece_cost_fn
```

... continued

... continued

```
 LOW VOL PLAIN SEATS(272).prod_vol = #method prod vol fn
  VOLCOEFFS(264).per_product[0]   1
  PRODUCT A(13).sales_volume = #uncertain TriangleFloat(500.000000,5000.000000,10000.000000)
  VOLCOEFFS(264).per_product[1] = 0
  VOLCOEFFS(264).per_product[2] = 0
 BYVOLIAO(252).prod_vols[0] = 4000.000000
 BYVOLIAO(252).prod_vols[1] = 7000.000000
 LOW VOL PLAIN SEATS(272).piece_cost - #uncertain TriangleFloat(200.000000,250.000000,300.000000)
 MED VOL PLAIN SEATS(273).piece_cost = #uncertain TriangleFloat(160.000000,200.000000,240.000000)
 HIGH VOL PLAIN SEATS(274).piece_cost = #uncertain TriangleFloat(140.000000,150.000000,200.000000)
PHYSICO(26).cardinalities[2] = 1
INTERIORTRIM(18).prod_vol = #method prod_vol_fn
 VOLCOEFFS(282).per_product[0] = 1
 PRODUCT A(13).sales_volume = #uncertain TriangleFloat(500.000000,5000.000000,10000.000000)
 VOLCOEFFS(282).per_product[1] = 1
 PRODUCT B(1).sales_volume = #uncertain TriangleFloat(500.000000,20000.000000,35000.000000)
 VOLCOEFFS(282).per_product[2] = 1
 PRODUCT C(2).sales_volume = #uncertain TriangleFloat(500.000000,10000.000000,12000.000000)
BYVOLIAO(86).prod_vols[0] = 25000.000000
BYVOLIAO(86).prod_vols[1] = 35000.000000
LOW VOL SEATBELTS(255).piece_cost = #uncertain TriangleFloat(37.500000,40.000000,41.900000)
MED VOL SEATBELTS(256).piece_cost = 25.000000
HIGH VOL SEATBELTS(257).piece_cost - 22.000000
PHYSICO(26).cardinalities[3] = 1
HEATING(87).piece_cost   #method piece_cost_fn
// ... HEATING has children in BOM tree
PHYSICO(26).cardinalities[4] = 1
FACIAASSY(88).piece_cost = #method piece_cost_fn
// ... FACIAASSY has children in BOM tree
PHYSICO(26).cardinalities[5] = 1
```

Appendix K: Design of RiTo

Contents

K.1 Introduction

This Appendix provides an overview of the design of the software toolkit which was built to support the methodology described in Chapter 8. The design is presented using the Fusion methodology which was described in Chapter 6. The "system" under design is SIMDLL.DLL as described in Section 10.1 of Chapter 10. It is this DLL which provides the core functionality for the toolkit. The Appendix concentrates on the key functions of object persistence, run-time schema querying (Section K.2) and the evaluation algorithm (i.e. the Monte Carlo engine, Section K.3). The class-based interface between the user interface (RITO.EXE) and SIMDLL.DLL, is outlined in Section K.4.

Some of the design and implementation issues which arose when building RiTo are described and some of the major design decisions which were taken are explained.

K.2 Object Persistence and Run-Time Schema Querying

It is necessary that the `Sim` objects comprising the risk model can be stored in a database and retrieved from it - i.e. they must be persistent. It must be possible to store to the database either the whole risk model, or just the currently live routes, or just the expected values, or just the most likely values - these last three being provided as an aid to model building. A second key requirement for all the `Sim` objects in the risk model is that they can provide information about their own class definitions, answering questions such as: what is the name of the class for this object? is it derived (directly or indirectly) from class X? how many attributes does it have? and links? what are their names? what are their types? is it legal within the schema to assign this pointer to that link? The response to such schema queries must be resolved at run-time, not at compile-time since RITO.EXE (the user-interface application) must be class-neutral, and must function correctly without recompilation when the schema is modified and user-defined classes are added or changed. Neither of these two key requirements are provided automatically by the C++ programming language.

As mentioned in Section 10.1 of Chapter 10, each class which is used by the designers to build the risk model (whether a base class, or user-defined) is represented by a single C++ class of the same name. The `Editable` C++ class provides both object persistence and run-time schema querying; the base and user-defined classes are all indirectly derived from `Editable` as described in Chapter 8, and thus inherit persistence and schema query capability. Therefore, the remainder of this section (Section K.2) describes the design of the `Editable` C++ class, and related classes, with the aid of Fusion notation diagrams and schemata.

K.2.1 Analysis

A `Model` consists of a network of `Editable` objects, with links between them. The `Model` contains a single "top object", which is an `Editable` object from which all others may be reached by following links recursively. Each `Editable` object has links to zero or more other such objects, and also contains zero or more attribute values - these may be of type FLOAT (floating point), BOOL (Boolean), INTEGER (integer) or Str (text string). Each `Editable` object is an instance of a class, whose description (class definition) is given by `ClassInfo` (i.e. `ClassInfo` is a meta-class). The complete set of classes which may be used within a `Model` is given by the `Schema`. The `Schema` contains a single "root class", which is the instance of `ClassInfo` from which all others are derived, directly or indirectly, in a single inheritance hierarchy. Thus each `ClassInfo` (except for the root class) has a single base-class and each has zero or more child classes. Each `ClassInfo` contains definitions of the methods, attributes and links which belong to the class - `LinkInfo`, `AttInfo` and `MethodInfo` respectively.

The system must be able to write a `Model` to a `Repository` (in the final implementation this was a human-readable ASCII file stored on disk), and read a `Model` from a `Repository`. At a given moment in time a `Model` will be engaged in one of several possible `Activities`, including "writing the entire model", "writing live routes", "writing most likely values", "writing expected values", "simulating" and "browsing". The system must also be able to retrieve a `Schema` from a schema file. However, the writing of the class definitions to the schema file is not part of the system, since this is handled by the Fusion CASE tool as described in Section 10.1 of Chapter 10.

The system must be able to supply all schema and instance information about `Editable` objects in the `Model`, in response to queries. In particular, the system must be able to return the `ClassInfo` for a specified `Editable` object and answer whether or not a specified `Editable` object "is a kind of" a specified `ClassInfo` (i.e. is the class of the `Editable` object either equal to the specified `ClassInfo` or derived from it). In order for RiTo to display the link tree (described in Section 10.2 of Chapter 10) , the system must be able to give the number of links from an `Editable` object, and then iterate through the links giving the name, class and cardinality of each link, as well as the value/s of the link (the identifier of the `Editable` object/s to which it points). Similarly, the number of attributes and the name, type, cardinality and value/s of each attribute must be made available in an iteration. There are many other schema queries which the system must support, but these have been omitted from the following Fusion model in the interest of brevity.

K.2.1.1 Fusion Object Model

Figure K-1 shows part of a Fusion object model, illustrating the relationships between the main classes involved in providing object persistence and supporting schema queries. The system boundary is shown as a dotted line. The main classes which lie within the system boundary are summarised in an excerpt from the data dictionary shown in Table K-1. The only agents (i.e. objects outside the system boundary) are the user interface and the schema file.

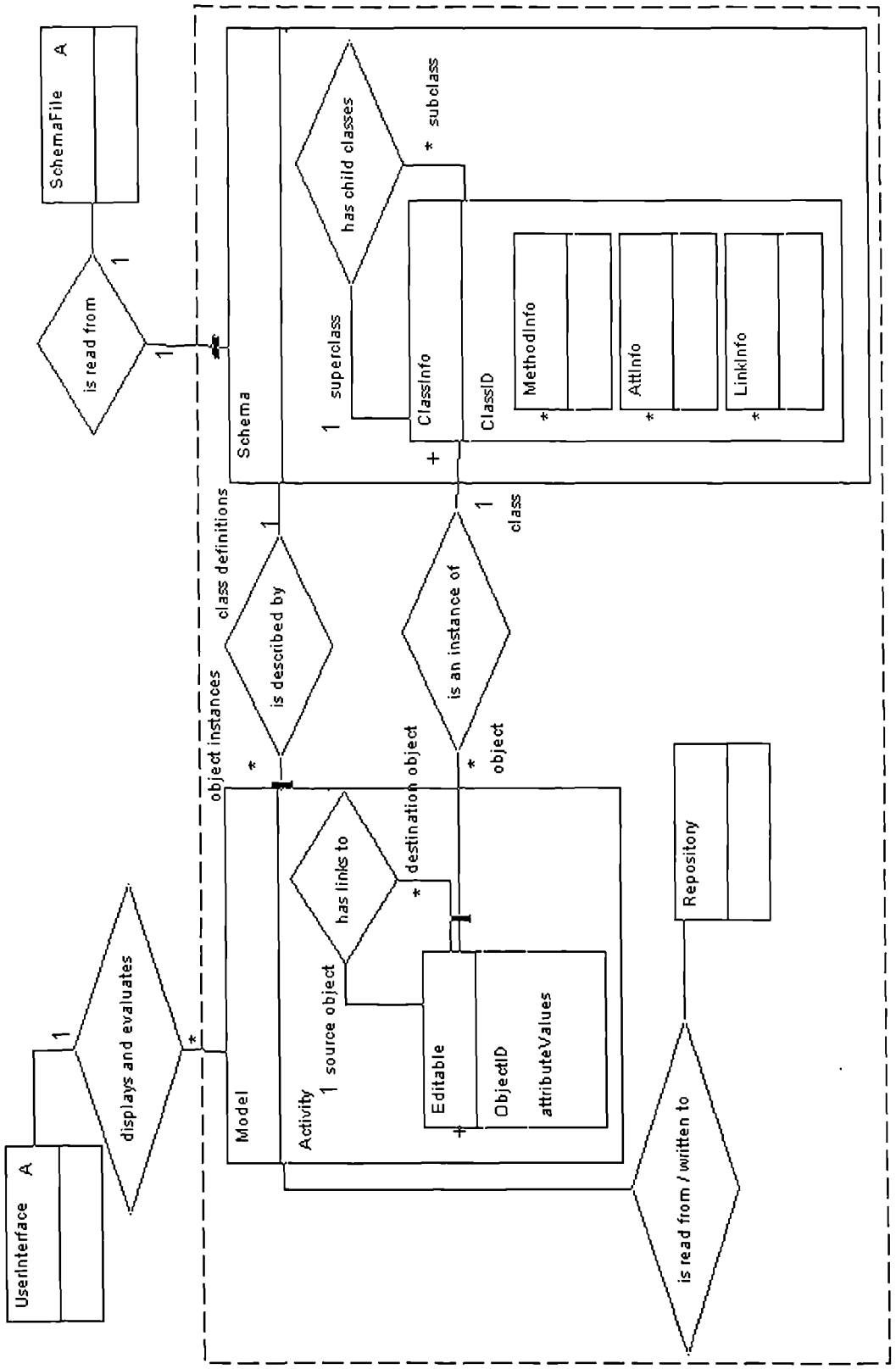


Figure K-1: Fusion object model showing Model and Schema

Class Name	Description
`Model`	A network of `Editable` objects.
`Editable`	An object which is capable of persistence and of responding to schema-queries. An `Editable` object is an instance of a class described by `ClassInfo` (an `Editable` object means an object of class `Editable` or derived class).
`Schema`	The set of all class definitions for a `Model`. A single inheritance hierarchy of `ClassInfo` objects.
`ClassInfo`	The meta-class. Each instance of `ClassInfo` describes a class which is either equal to, or derived from, the class `Editable`. Each `Editable` object is described by an instance of `ClassInfo`.
`Repository`	The persistent storage for a `Model`.
`MethodInfo`	A description of a method - this must include the name and output type. Output type may be FLOAT/INTEGER/BOOL or Str.
`AttInfo`	A description of an attribute - its name, type (FLOAT/INTEGER/BOOL/Str), cardinality (which may be any fixed value or may be defined as variable), and whether or not it is read-only.
`LinkInfo`	A description of a link - its name, class, whether or not it is bound, whether or not it is shared, and whether or not it is read-only.
`SchemaFile`	A file containing all the class definitions for a `Schema`. An instance of `Schema` can be created from such a file. The class descriptions are stored in the file in Fusion class definition syntax.

Table K-1: Excerpt from data dictionary showing classes involved in persistence and schema queries

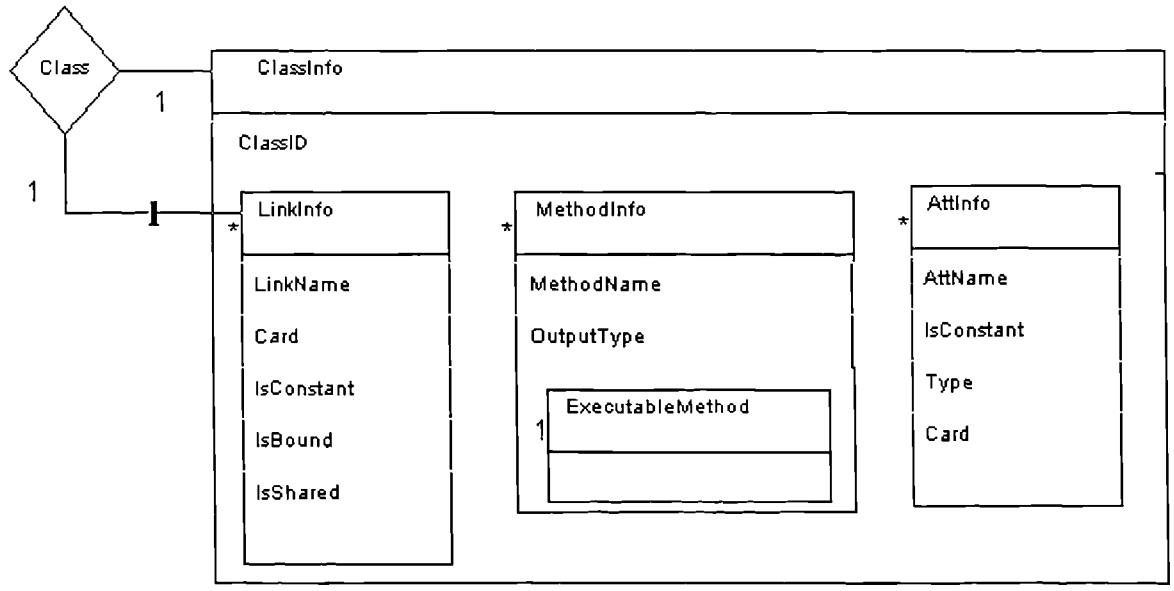


Figure K-2: Fusion object model showing Schema detail

Figure K-2 shows `MethodInfo`, `AttInfo` and `LinkInfo` in more detail. The information stored regarding a method must include its name and also the data type of the value which it returns - the principle of polymorphism means that it should be permitted for a method returning a FLOAT for example, to be used as a derivation route for an attribute value of type BOOL, but the user should nevertheless be aware that such a conversion is taking place since information is being lost. There must also be some mechanism for invoking a method from its `MethodInfo` description - this is simply represented in Figure K-2 as a sub-object called `ExecutableMethod`. However, the decision over whether or not the method should be explicitly represented as part of `MethodInfo` (for example as an algebraic expression stored in a string) was not made during this, first, "analysis" stage of the Fusion methodology, and is thus not discussed further here.

The class definitions are stored in the schema file in Fusion class definition notation. Therefore, the information stored regarding an attribute must include its name, its mutability (constant / variable), its data type and its cardinality - all shown as attributes in Figure K-2. The information stored regarding a link must include its name and cardinality, its mutability (constant / variable), its binding (bound / unbound) and its

exclusivity (shared / exclusive). This information is shown as attributes in Figure K-2, and the class of the link is indicated by its relationship to an instance of `ClassInfo`.

K.2.1.2 Fusion Interface Model

Part of the system context diagram is shown in Figure K-3, and an example time-line scenario is shown in Figure K-4.

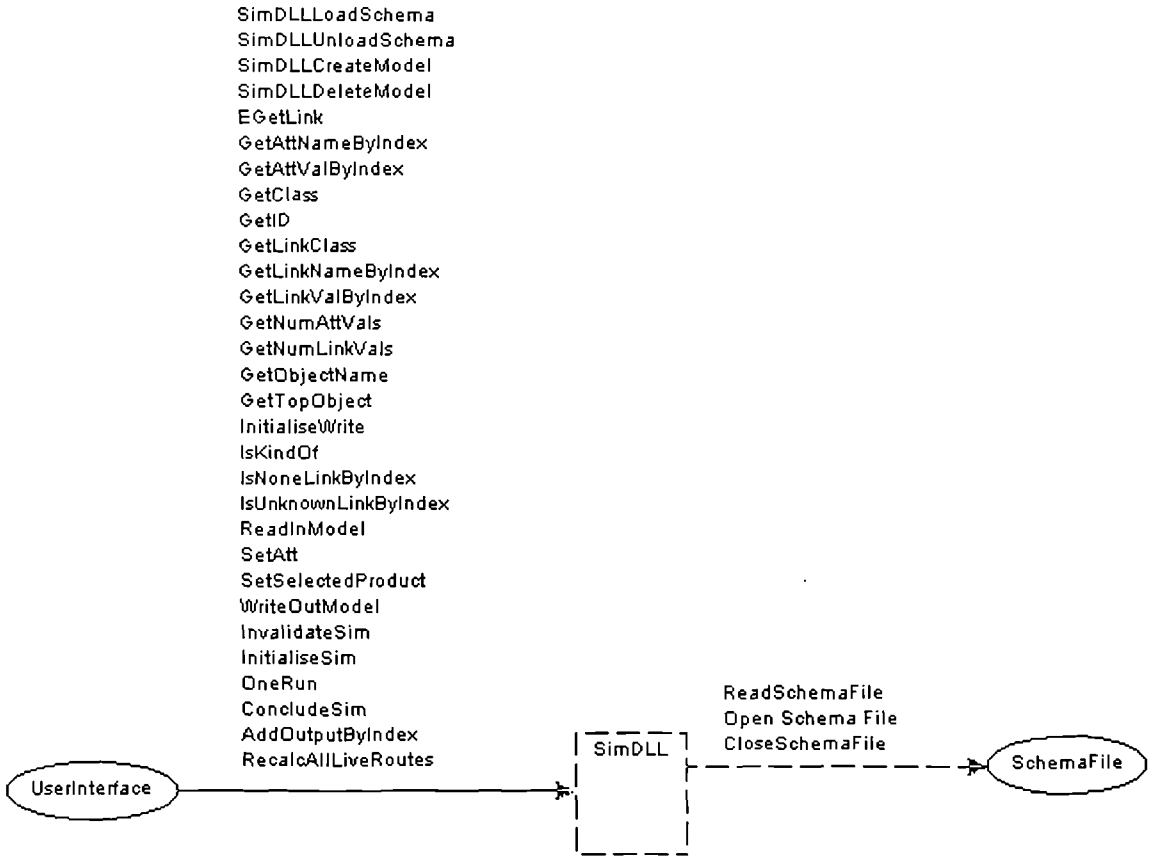


Figure K-3: Part of system context diagram

Figure K-4 shows the sequence of calls which would be made by the user interface software if it simply began execution, opened an existing risk model, displayed the top object in the link tree and then completed execution. Notice that no output events are shown for the "`Get<xxx>`" system operations; strictly speaking, in the Fusion method, an output event should be defined for each such system operation. For example, after the `GetTopObject` system operation, an output event should be defined which sends the identity of the top object to the user interface. The life-cycle model should then constrain this output event to follow the `GetTopObject` system operation. However, in the work described here this would introduce an unnecessary level of complexity into the Fusion model - the Agent is another piece of software, and the communication is not asynchronous. It is clearly simpler and clearer in this situation to use the return value from a system operation to represent the output event which supplies the requested information back to the calling code.

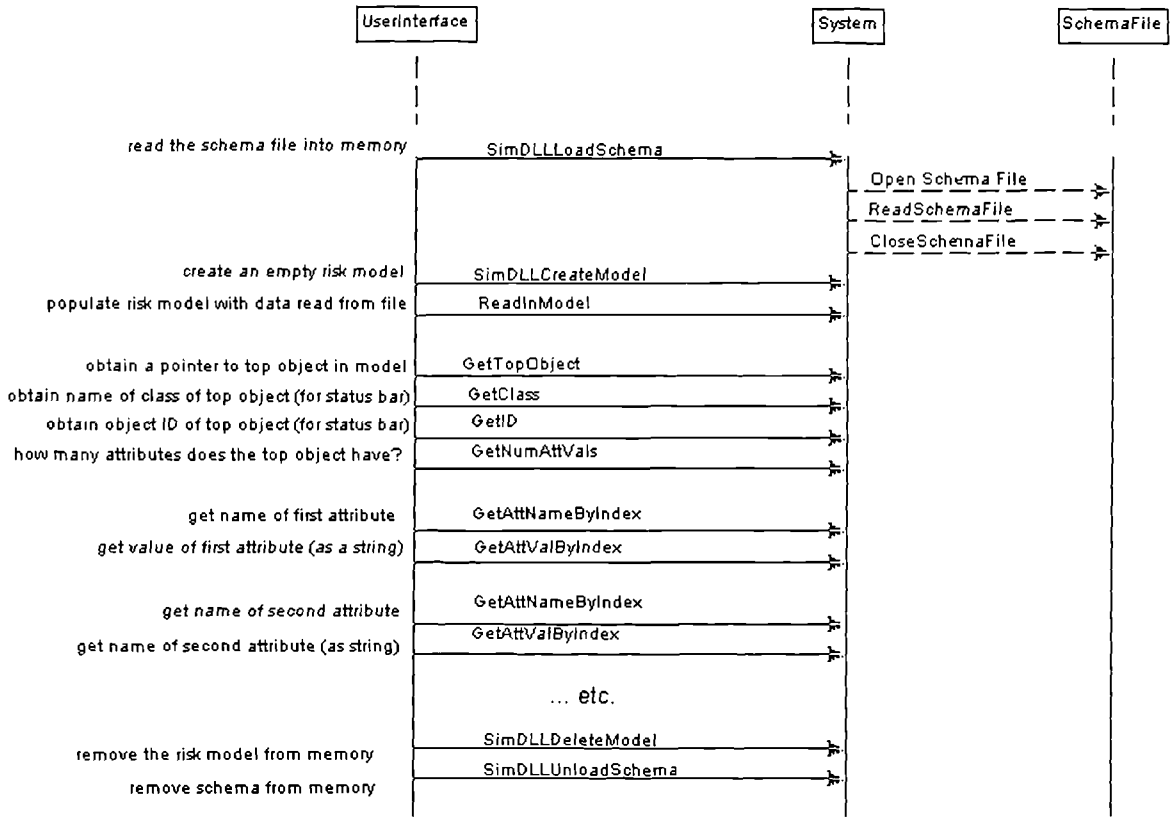


Figure K-4: Example time-line scenario for displaying top object in link tree

Part of the lifecycle model for SIMDLL is shown below - only those scenarios concerned with object persistence and run-time schema queries are shown here:

```
lifecycle SimDLL : SimDLLLoadSchema .
                   SchemaFile : # Open Schema File .
                   SchemaFile : # ReadSchemaFile .
                   SchemaFile : # CloseSchemaFile
                   ( EditARiskModel * ) || ( EditARiskModel * ) . SimDLLUnloadSchema

EditARiskModel = SimDLLCreateModel . [ ReadInModel ] . GetTopObject (
                 SelectObjectInFullLinkTree |
                 OpenBranchInFullLinkTree |
                 SelectObjectInBOMLinkTree |
                 OpenBranchInBOMLinkTree |
                 AddOuptut |
                 DeleteOutput |
                 Simulate |
                 DisplayGraph |
                 DisplayStatistics |
                 SetSelectedProduct |
                 ToggleCustomerOption |
                 CalculateProductionVolumes |
                 SaveFile
                 ) * . SimDLLDeleteModel

DisplayAttributeList = GetNumAttVals . ( GetAttNameByIndex . GetAttValByIndex ) *

SelectObjectInFullLinkTree = GetClass . GetID . DisplayAttributeList

OpenBranchInFullLinkTree = GetNumLinkVals . (
                           GetLinkNameByIndex .
                           IsNoneLinkByIndex . IsUnknownLinkByIndex [
                           GetLinkValByIndex . (
                           IsKindOf * . GetObjectName . GetNumLinks .
                           ( GetLinkClass . IsKindOf ) * . [ EGetLink . [ EGetLink ] ]
                           ) *
                           ]
                           ) *

SetSelectedProduct = SetSelectedProduct

ToggleCustomerOption = SetAtt

SaveFile = InitialiseWrite . WriteOutModel
```

The lifecycle model illustrates a modelling difficulty which arose during the course of this work; the system must allow multiple risk models to be opened and edited simultaneously (because RiTo has a multiple document interface). This cannot be shown using the Fusion lifecycle syntax - the model above attempts to convey the correct impression by showing zero or more `EditARiskModel` scenarios arbitarily interleaved with zero or more `EditARiskModel` scenarios.

In the scenario entitled `SelectObjectInFullLinkTree`, the name of the object class and the object ID are obtained in order that they may be displayed in the status bar at the bottom of the link tree, and then the names and values of each the attributes are obtained for display in the "Attributes of Selected Object" area of the screen. A simple example of the `OpenBranchInFullLinkTree` scenario is illustrated in Figure K-5:

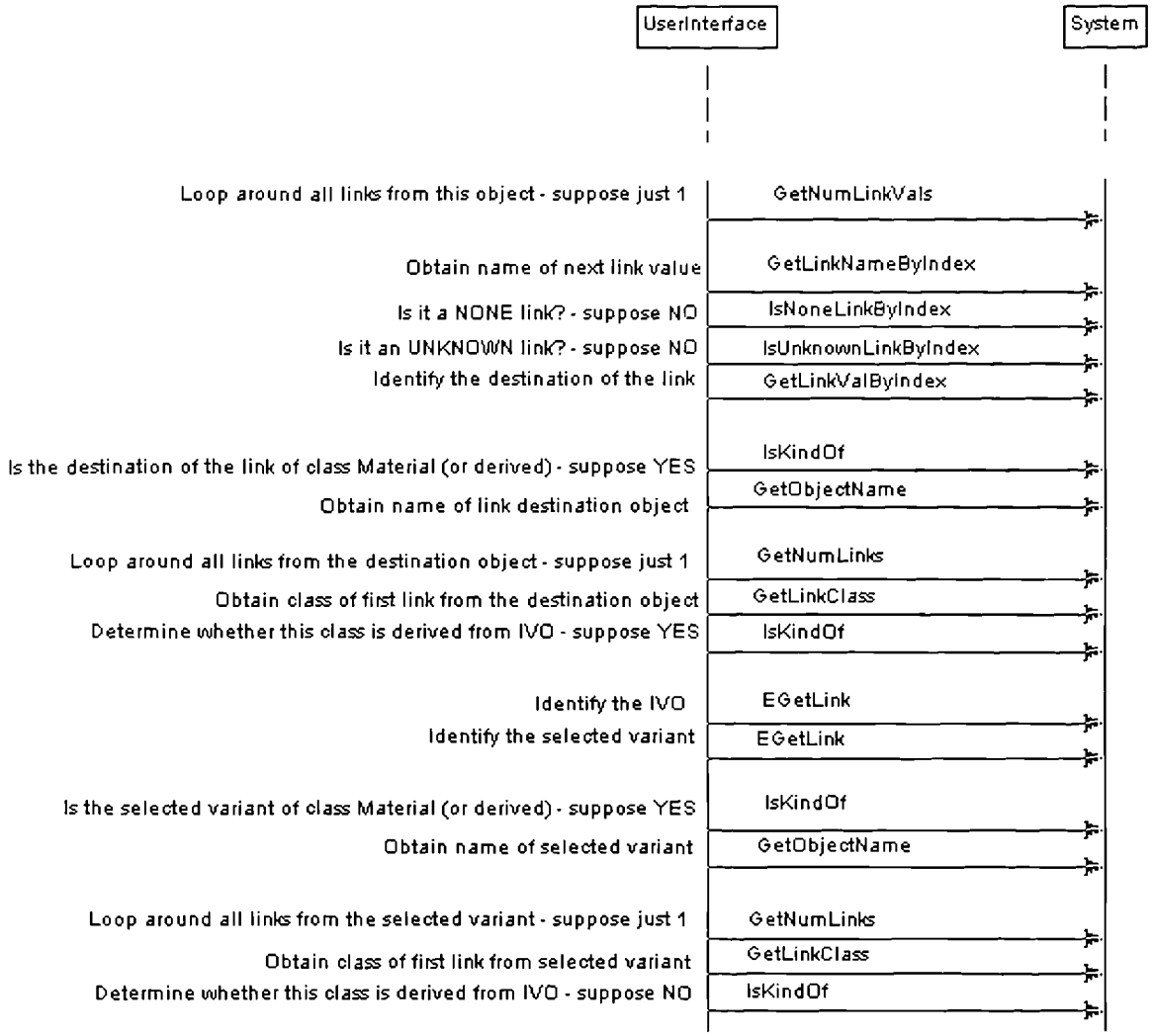


Figure K-5: Example time-line scenario for opening a branch in the full link tree

If a link is neither Unknown nor None, then the `Editable` object to which it points is retrieved and then there are repeated invocations of the `IsKindOf` system operation - needed to determine which of the available icons should be used to represent the object in the link tree. Then, having determined the name of the `Editable` object, there are repeated calls to `GetLinkClass` and `IsKindOf` which are needed to determine whether the `Editable` object has variants - does it have a link to an `IVO` object? If it transpires that the `Editable` object does have variants, then two optional calls to `EGetLink` (the name signifies `Editable GetLink`) are required to retrieve the `IVO` and then to retrieve the selected object for that `IVO`. The scenarios for selecting objects and opening branches in the BoM view of the link tree are similar but more complex, and are not shown here - but they use the same set of system operations.

Having built the system context diagram, time-line scenarios and lifecycle model, the remaining stage in the development of the Fusion analysis interface model is building the operation model. The Fusion operation model schemata for three examples of system operations which make no modification to the system (`GetLinkNameByIndex`, `GetAttValByIndex` and `IsKindOf`) are shown below:

```
Operation : GetLinkNameByIndex

Supplied : editable_object : Editable ,
           i : INDEX

Returns : Str

Description : " Returns the name of the i'th link value which has
editable_object as its source.  The name of the link has the string
"[<card>]" appended to it if the cardinality, <card>, is greater than 1.
i is index into all link values from editable_object i.e. a single link
with cardinality of C is counted as C link values. "

Reads : editable_object : Editable ,
        classInfo : ClassInfo with " class of editable_object ==
                    classInfo "

Assumes : " editable_object has at least i + 1 link values "

Result : " Unchanged "
```

```
Operation : GetAttValByIndex

Supplied : editable_object : Editable ,
           i : INDEX

Returns : Str

Description : " Returns the value of that attribute of editable_object
with index i. The value is expressed as a string and thus the same system
operation can be used for all attribute types. "

Reads : editable_object : Editable ,
        classInfo : ClassInfo with " class of editable_object ==
                    classInfo "

Assumes : " editable_object has at least i + 1 attribute values "

Result : " Unchanged "
```

```
Operation : IsKindOf

Supplied : parent : ClassInfo ,
           editable_object : Editable

Returns : BOOL

Description : " Returns TRUE if editable_object is a member of class
parent, or of one of its sub-classes.  Otherwise, returns FALSE. "

Reads : classInfo : ClassInfo with " class of editable_object
                                   == classInfo " ,
        parent : ClassInfo

Result : " Unchanged "
```

Each link which is defined in the schema is identified by a link identifier - this number is unique within a given `Editable` object, and is used to enumerate the links from an `Editable` object or within a given class. A typical `Editable` object, say of class X, will have some links defined locally in class A, some from class B and so on , where A and B are super-classes of X. The link identifiers are assigned such that the lowest ID (0) belongs to the first link defined locally in the "highest" parent class in the inheritance hierarchy (i.e. the most distant relative, closest to the root class). Since we have no multiple inheritance, the ID of a link will remain the same whether it occurs as part of an object of class A, class B or class X. Link identifiers are of type LINKNUM. A similar definition is used for the attribute identifiers, which are of type ATTNUM.

The index of a link value (e.g. the second argument to `GetLinkNameByIndex` above) is not the same as the identifier of the link. Some links will have a cardinality greater than 1, and some will have variable cardinality. Thus the number of link values (pointers to other `Editable` objects) will generally differ from

the number of links (with unique link definition in the schema). The index is used to enumerate the link values (or attribute values) for a specified `Editable` object.

The two system operations involved in storing the risk model to the repository, and the one involved in reading the risk model from the repository, provide examples of system operations which change the state of the system:

```
Operation : InitialiseWrite

Supplied : model_object : Model ,
           state : Activity

Description : " Prepares model_object for writing to the repository. The
value of state may be WRITING_ALL (if whole model is to be written),
WRITING_LIVE_ROUTES (if live routes only are to be written), or
WRITING_INDICATIVE_VALUES (if expected values or most likely values are
to be written). "

Changes : model_object : Model ,
          model_object : Model . Activity : Activity

Result : " model_object.Activity has been assigned a value of state.
All necessary initialisation of flags has been carried out. "
```

```
Operation : WriteOutModel

Supplied : pathName : Str ,
           model_object : Model

Returns : ErrorResultType

Description : " Creates a new object repository associated with a file
with path pathName, overwriting any previously existing file with the
same path name. Stores model_object in the new object repository by
writing the top object of model_object and then writing each Editable
object which can be reached by recursively traversing links from the top
object. "

Reads : model_object : Model . Activity : Activity,
        top_object : Editable with " top_object == the top object in
                     model_object "

Changes : new repository : Repository

Assumes : " pathName is a valid path. "

Result : " model_object has been stored to pathName. "
Operation : ReadInModel

Supplied : model_object : Model ,
           PathName : Str ,
           TopObjectID : OBJECT_ID

Returns : ErrorResultType

Description : " Populates model_object using the object repository stored
in file with path PathName. The top object (searched for first in the
repository) has object ID TopObjectID. "

Changes : model_object : Model ,
          new repository : Repository

Assumes : " model_object is empty.
A file exists with path PathName, containing a valid repository.
An Editable object exists in the repository with ID TopObjectID. "

Result : " All Editable objects which may be reached by recursively
following links from the top object have been created in model_object and
populated with values read from the repository. "
```

The remaining operation model schemata have been omitted in the interests of brevity. This concludes the analysis phase of the Fusion method.

K.2.2 Design

K.2.2.1 Fusion Object Interaction Graphs

One of the simplest system operations is `IsKindOf` - Figure K-6 shows its object interaction graph (OIG):

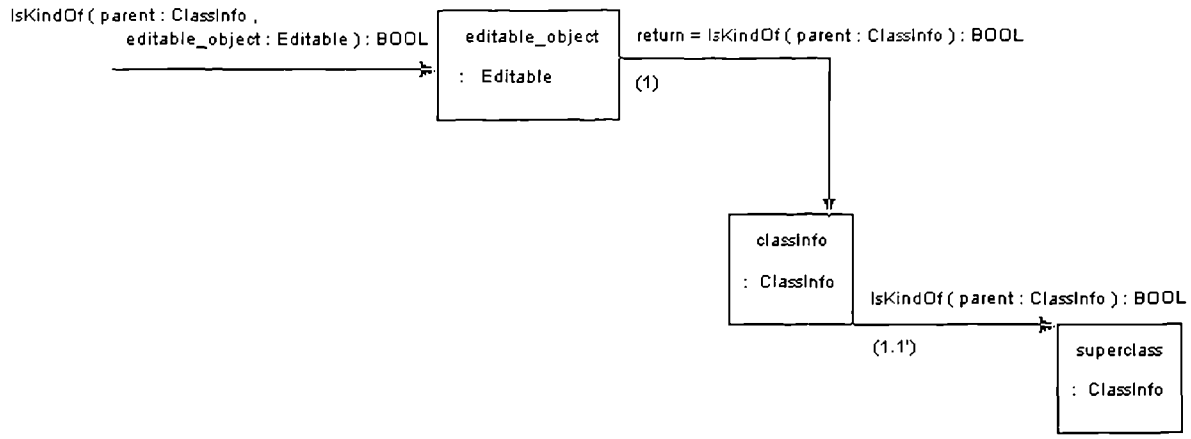


Figure K-6: Object interaction graph (OIG) for the IsKindOf system operation

The accompanying description (which should be stored in the data dictionary) is:

```
operation BOOL Editable::IsKindOf(parent:ClassInfo)

/* Determines whether this object is a member of class parent or of a
derived class */

Look up classInfo, the class of this Editable object, and ask whether
this is a kind of parent.
```

```
method BOOL ClassInfo:IsKindOf(parent:ClassInfo)

/* Determines whether this class is equal to class parent or to a
derived class */

IF this ClassInfo object == parent THEN return TRUE.

IF this ClassInfo object has no base class THEN return FALSE.

Look up superclass, the base class of this ClassInfo object, and ask
whether this is a kind of parent.
```

This shows how the single inheritance hierarchy stored in the schema is used to respond to such run-time queries, with each `ClassInfo` object checking its superclass, recursively, until a match for the class of interest is found or the root class (`Editable`) is reached. Another simple system operation is `GetAttValByIndex;` the OIG is shown in Figure K-7 and the description is shown below:

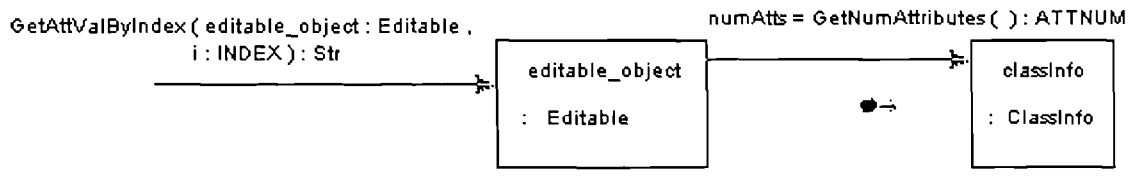


Figure K-7: OIG for the GetValByIndex system operation

```
operation Str Editable::GetAttValByIndex(INDEX i)

/* Returns the attribute value with index i as a string. (NB Index i is
incremented by C for an attribute with cardinality of C). */

Look up classInfo, the class of this Editable object, and obtain the
number of attributes, numAtts.

Calculate the attribute identifier (attNum) and cardinality (card)
corresponding to index i.

Copy the value of attribute attNum, cardinality card, into a string s.
Return s.
```

A separate sub-OIG is created for `ClassInfo::GetNumAttributes` (shown in Figure K-8):

```
method ATTNUM ClassInfo::GetNumAttributes()

/* Returns the total number of attributes for this class, including
inherited attributes. */

IF this ClassInfo object is the root class (Editable) THEN return the
number of local attributes.

Look up superclass, the base class of this ClassInfo object.

Return superclass.GetNumAttributes() + the number of local attributes.
```

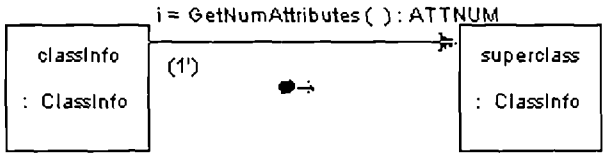


Figure K-8: OIG for the method ClassInfo::GetNumAttributes

The OIG for `GetLinkNameByIndex` is shown in Figure K-9 and the description is below. (`ClassInfo::GetNumLinks` is analogous to `ClassInfo::GetNumAttributes` and so is not described further). The mechanism for obtaining information about the link with identifier `linkNum` is to count back through the super-classes until the `ClassInfo` is reached which has `linkNum` as one of its local links - this is achieved by `ClassInfo::GetLinkInfo`, below.

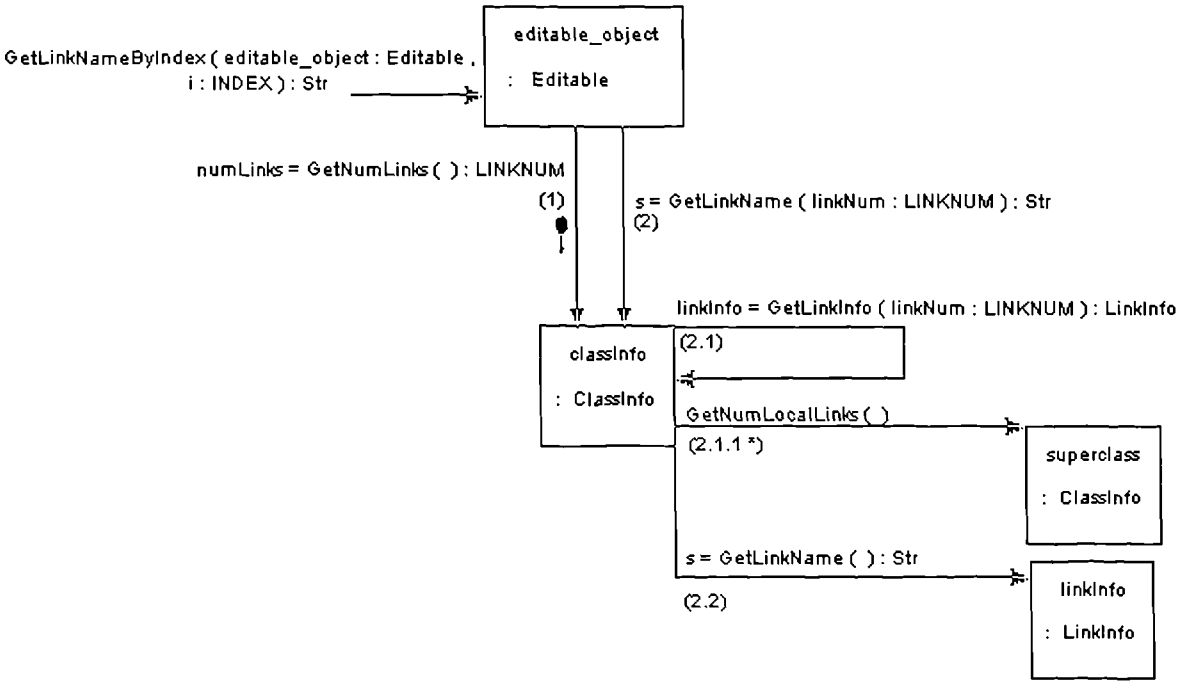


Figure K-9: OIG for the system operation GetLinkNameByIndex

```
operation Str Editable::GetLinkNameByIndex(INDEX i)

/* Returns the name of the link with index i as a string. The name of the
link has the string "[<card>]" appended to it if the cardinality, <card>,
is greater than 1. */

(1) Look up classInfo, the class of this Editable object, and obtain the
    number of links, numLinks.

    Calculate the link identifier (linkNum) and cardinality (card)
    corresponding to index i.

(2) Obtain the name of link linkNum in classInfo and copy it into a
    string s.

     (2.1) Obtain the linkInfo with identifier linkNum by invoking

           GetLinkInfo on classInfo.

     (2.2) Obtain the name of the identified linkInfo.

    IF (card > 1) THEN Append "[card]" to s.

    Return s.
```

```
method LinkInfo ClassInfo::GetLinkInfo(LINKNUM linkNum)

/* Returns information describing the linkNum'th link  belonging to this
class, including inherited links.*/

Obtain the number of links, including inherited links, numLinks.
countBack = numLinks - linkNum - 1.

i = 0.

superClass = this ClassInfo object.

WHILE (thisClass exists)

      i = i + number of local links in superClass.

      IF i > countBack THEN
            return local link (i - countBack - 1) of superClass
      ENDIF

      superClass = the superclass of superClass.

ENDWHILE
```

Having described the system operations for run-time schema querying, we now move on to those which provide object persistence. The OIGs for `InitialiseWrite` and `WriteOutModel` are illustrated in Figure K-10 and Figure K-11.

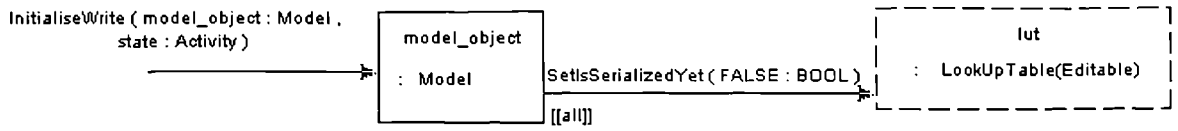


Figure K-10: OIG for system operation InitialiseWrite

A look-up table is used to store the collection of all `Editable` objects in a model, with the index into the table equal to the object ID. Both reading-in and writing-out of the model is referred to as "serialization" (following the Microsoft Foundation Class convention), since the information is stored serially in the repository. Every `Editable` object in the look-up table has a flag which is used to indicate whether or not it has been serialized yet. These are initially all assigned a value of FALSE, as shown in Figure K-10. In Figure K-11, the first argument to the constructor for the repository is TRUE - this assigns a value of TRUE to `Repository::IsStoring`, indicating that the repository will be used to write out a risk model (a

value of FALSE would indicate that it would be used to read in a model). The object interaction graph illustrated for `Editable::Serialize` describes the situation when `Repository::IsStoring` is TRUE - a similar approach is taken to reading in a risk model, but this is not illustrated here. The description for `Editable::Serialize` is given below. Notice that a `ClassInfo` object is stored for each of the four base types (INTEGER/FLOAT/BOOL/Str) as well as for each `Editable` class, and one of these four is returned by `ClassInfo::GetAttributeClass`.

The methods `Editable::GetAtt()` and `Editable::GetLink()`, mentioned in the method description for `Editable::Serialize()`, are overridden in `Sim` to provide a choice of writing live route, all routes or indicative value only, according to the value of `Model.Activity`. The versions of these methods provided in `Editable` simply write point values (object ID or attribute value string) or "NONE" or "UNKNOWN".

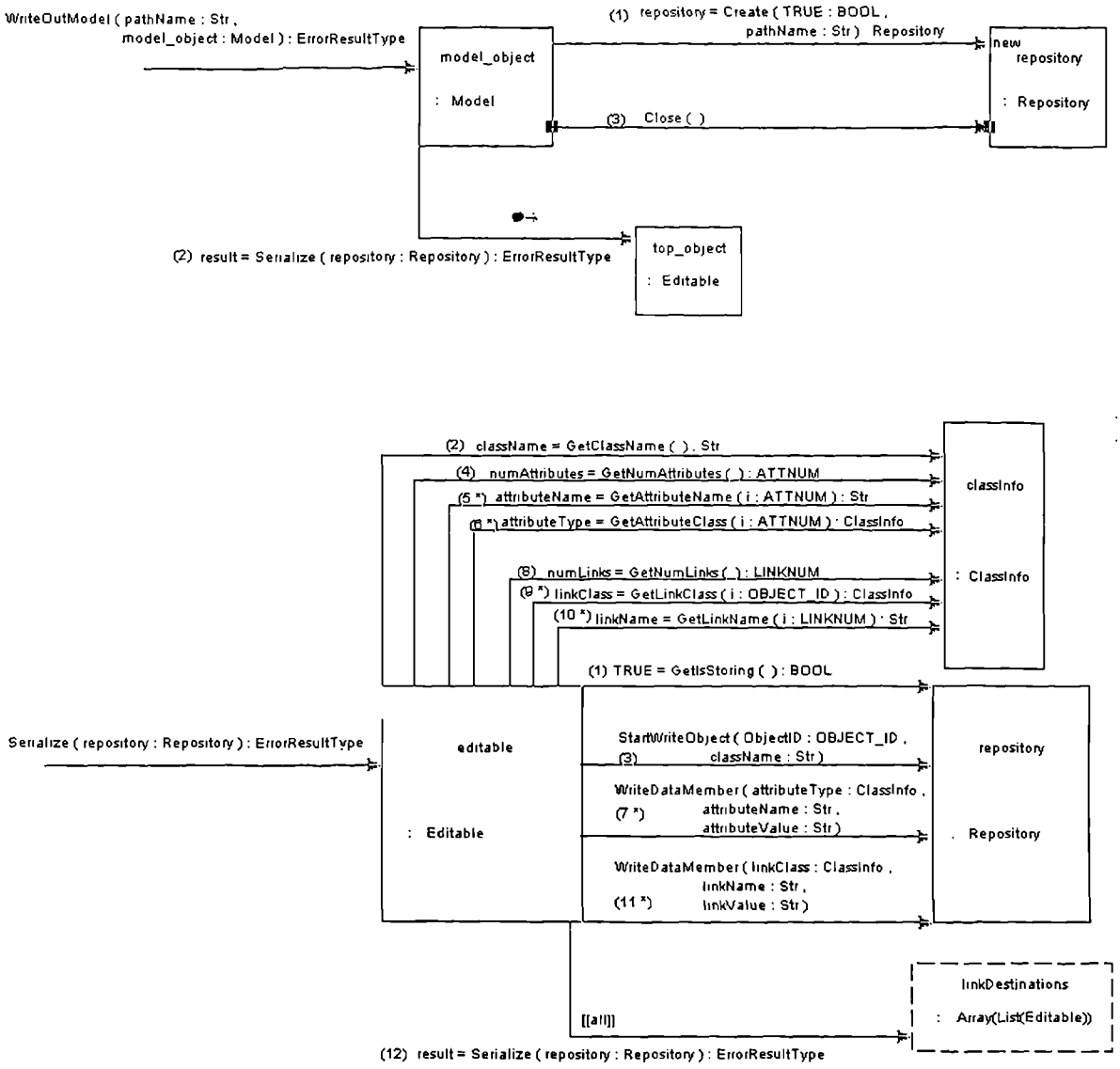


Figure K-11: OIGs for system operation WriteOutModel and for method Editable::Serialize when IsStoring == TRUE.

```
method ErrorResultType Editable::Serialize(Repository repository)

/* Stores this Editable object into repository (or retrieves it from
repository), along with all other Editable objects to which it has links.
*/

    IF (IsSerializedYet == TRUE) THEN return OK.

    IsSerializedYet = TRUE.

(1) Obtain repository.IsStoring

    IF (repository.IsStoring == TRUE) THEN

     (2) Obtain the name of the class of this Editable object.

     (3) Write the object ID and class name to the repository.

     (4) Obtain the number of attributes including inherited ones,
         numAttributes.

         FOR i = 0 TO numAttributes

          (5) Obtain name of i'th attribute, attributeName.

          (6) Obtain type of i'th attribute (INTEGER/FLOAT/BOOL/Str),
              attributeType.

              CALL GetAtt() to obtain val, the value of the i'th
              attribute expressed as a string of the form "v1,..,vn"
              where n is the cardinality of the attribute.

          (7) Write attributeName, attributeType and val to the
              repository.

         NEXT i

     (8) Obtain the number of links including inherited ones,
         numLinks.

         FOR i = 0 TO numLinks

          (9)  Obtain class of i'th link, linkClass.

          (10) Obtain name of i'th link, linkName.

               FOR j = 0 TO cardinality of i'th link

                IF j'th value of i'th link is not NONE or UNKNOWN THEN
                 Add j'th value of i'th link to linkDestinations
                ENDIF

               NEXT j

               CALL GetLink() to obtain val, the value of the i'th link
               expressed as a string of the form "id1,..,idn" where n is
               the cardinality of the link and idj is an object ID.

          (11) Write linkName, linkClass and val to the repository.

         NEXT i

    (12) Invoke Serialize() on each element of linkDestinations.

    ELSE
      ...// Pseudo-code for reading in an Editable object from
         // repository into memory
    ENDIF
```

K.2.2.2 Fusion Visibility Graphs

Visibility graphs for `ClassInfo`, `Editable` and `Model` are shown in Figure K-12, Figure K-13 and Figure K-14.

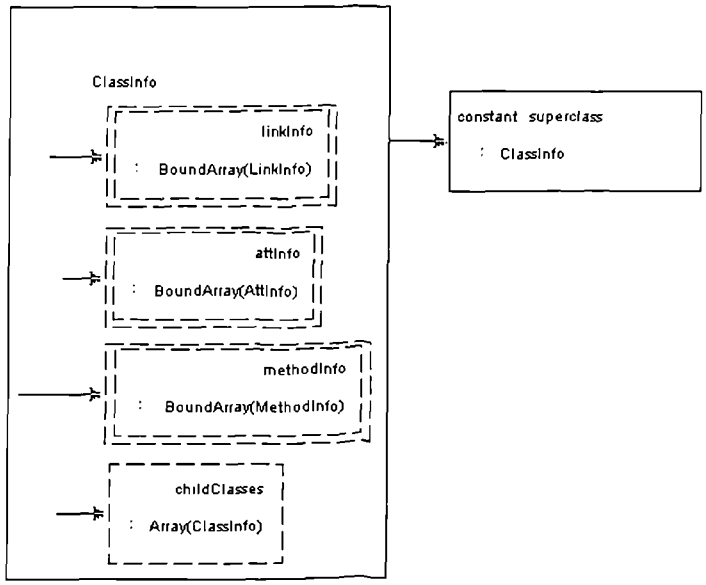


Figure K-12: Visibility graph for ClassInfo

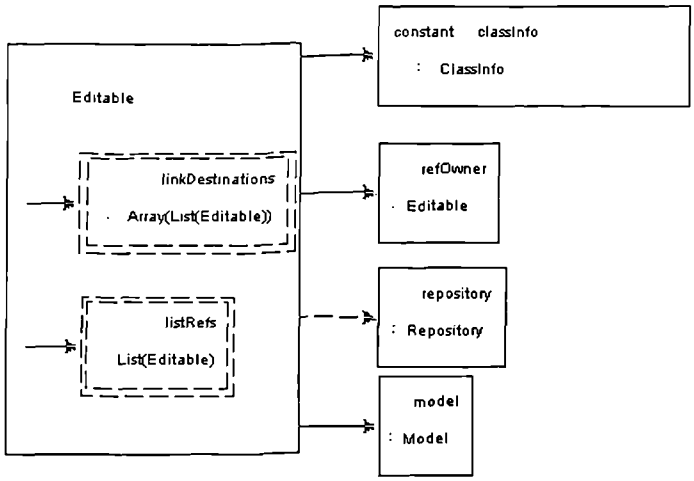


Figure K-13: Visibility graph for Editable

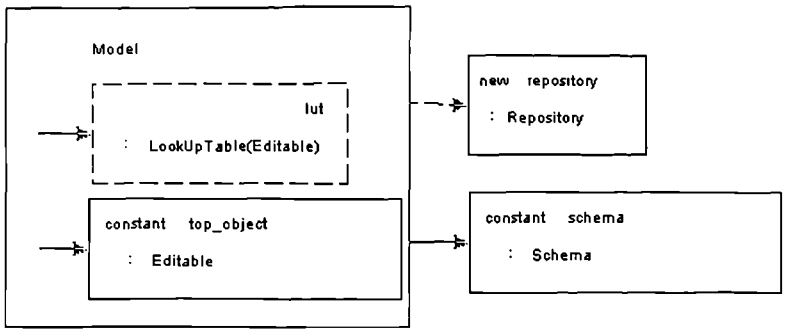


Figure K-14: Visibility graph for Model

Note that the properties of the links shown (e.g. binding and exclusivity) refer to the collection, not to its elements. Element binding is specified by the collection class - for example the elements of the `BoundArray` collection class are bound into the collection, unlike those in the `Array` collection class. Visibility links to `refOwner` and `listRefs` which are shown in Figure K-13 are used to record back references, necessary for automatic garbage collection - this is not described further here.

K.2.2.3 Fusion Class Descriptions and Implementation Notes

Descriptions of `ClassInfo` and associated classes are shown below. A decision was taken to avoid an explicit representation of the executable method in `MethodInfo`, and instead, for each method named in the schema, to define a C++ method on the corresponding C++ class (generated by the automatic code generator). Each C++ class has a single hard-coded method (also created by the code generator) which stores a function pointer (see Section 6.3.2 in Chapter 6) to each schema method into an array, so that the schema methods can be iterated over using an integer method identifier which indexes the array.

```
class MethodInfo isa
 attribute variable MethodName  : Str
 attribute constant outputType  : shared unbound ClassInfo
endclass
```

```
class AttInfo isa
 attribute variable AttName  : Str
 attribute variable IsConstant  : BOOL
 attribute variable Type  : BYTE
 attribute variable Card  : CARD
 method GetClass (  ) : ClassInfo
endclass
```

```
class LinkInfo isa
 attribute variable LinkName  : Str
 attribute variable Card  : CARD
 attribute variable IsConstant  : BOOL
 attribute variable IsBound  : BOOL
 attribute variable IsShared  : BOOL
 method GetLinkName (  ) : Str
 attribute variable linkClass  : shared unbound ClassInfo
endclass
```

```
class ClassInfo isa
 attribute variable ClassID  : OBJECT_ID
 attribute variable superclass  : shared unbound ClassInfo
 attribute variable childClasses  : shared bound col Array(ClassInfo)
 attribute variable linkInfo  : exclusive bound col BoundArray(LinkInfo)
 attribute variable attInfo  : exclusive bound col BoundArray(AttInfo)
 attribute variable methodInfo  : exclusive bound col
  BoundArray(MethodInfo)
 method IsKindOf ( parent : ClassInfo ) : BOOL
 method GetNumAttributes (  ) : ATTNUM
 method GetNumLinks (  ) : LINKNUM
 method GetLinkName ( i : LINKNUM ) : Str
 method GetLinkInfo ( linkNum : LINKNUM ) : LinkInfo
 method GetNumLocalLinks (  ) : LINKNUM
 method GetClassName (  ) : Str
 method GetAttributeName ( i : ATTNUM ) : Str
 method GetAttributeClass ( i : ATTNUM ) : ClassInfo
 method GetLinkClass ( i : LINKNUM )
 // ...
endclass
```

The class descriptions for `ClassInfo` (above) and `Editable` and `Model` (below) are incomplete, but show the attributes and methods described so far in this Appendix. `Editable.attributeValues` is shown below as having type `void **` (a C language construct for a pointer to a pointer of arbitrary type) - this is effectively equivalent to a collection of collections of arbitrary type, analogous to `Editable.linkDestinations`.

```
class Editable isa
 attribute variable ObjectID  : OBJECT_ID
 attribute variable attributeValues  : void * *
 attribute variable linkDestinations  : exclusive bound col
  Array(List(Editable))
 attribute variable linkCardinalities  : Array(CARD)
 attribute variable attributeCardinalities  : Array(CARD)
 attribute constant classInfo  : shared unbound ClassInfo
 attribute variable repository  : shared unbound Repository
 attribute constant model  : shared unbound Model
 attribute variable listRefs  : exclusive bound col List(Editable)
 attribute variable refOwner  : shared unbound Editable
 method GetObjectName ( ) : Str
 method IsKindOf ( parent : ClassInfo ) : BOOL
 method GetAttValByIndex ( i : INDEX ) : Str
 method GetLinkNameByIndex ( i : INDEX ) : Str
 method Serialize ( repository : Repository ) : ErrorResultType
 method EGetLink ( i : LINKNUM ,
                   w : CARD ) : Editable
 method GetAttNameByIndex ( i : INDEX ) : Str
 method GetClass ( ) : ClassInfo
 method GetID ( ) : OBJECT_ID
 method GetLinkClass ( i : LINKNUM )
 method GetLinkValByIndex ( i : INDEX ) : Editable
 method GetNumAttVals ( ) : INDEX
 method GetNumLinkVals ( ) : INDEX
 method GetNumLinks ( ) : LINKNUM
 method IsNoneLinkByIndex ( i : INDEX ) : BOOL
 method IsUnknownLinkByIndex ( i : INDEX ) : BOOL
 method SetAtt ( value : Str ,
                 A : ATTNUM ,
                 C : CARD )
 method FGetAtt ( A : ATTNUM ,
                  C : CARD ) : FLOAT
 method BGetAtt ( A : ATTNUM ,
                  C : CARD ) : BOOL
 method IGetAtt ( A : ATTNUM ,
                  C : CARD ) : INTEGER
 method SGetAtt ( A : ATTNUM ,
                  C : CARD ) : Str
// ...
endclass
```

```
class Model isa
 attribute variable Activity  : Activity
 attribute variable lut  : shared bound col LookUpTable(Editable)
 attribute constant top_object  : shared bound Editable
 attribute constant schema  : shared unbound Schema
 method SimDLLCreateModel ( ClassName : Str ,
                            TopObjectID : OBJECT_ID ) : Model
 method ReadInModel ( PathName : Str ,
                      TopObjectID : OBJECT_ID ) : ErrorResultType
 method InitialiseWrite ( state : Activity )
 method WriteOutModel ( pathName : Str ) : ErrorResultType
 method GetTopObject ( ) : Editable
 method SetSelectedProduct ( productIndex : CARD ) : ErrorResultType
 // ...
endclass
```

In the final implementation of the risk tool, the values of the attributes for all `Editable` and `Sim` classes are stored in `Editable`, as shown above, but static data members are defined in each derived C++ class representing the attribute and link identifiers. (In C++ a static data member has only one value stored, which is accessed by all instances of the class.) The C++ code defining these static data members and assigning them suitable values is generated by the automatic code generator. For example, a C++ class `PhysicalObject` (which is derived indirectly from `Sim`), has a static data member of type ATTNUM named `ATT_piece_cost`. This data member takes a value of 3, which is the attribute identifier for piece cost. The actual floating point value for the piece cost is stored in `Editable` as part of `Editable::attributeValues`, and methods defined on `PhysicalObject` can access the value by passing `ATT_piece_cost` as the first parameter to the method `Editable::FGetAtt()` (i.e. Floating-

point-Get-Attribute). This is faster than passing a string (e.g. "piece_cost") as the argument to `FGetAtt()`, which is important because this mechanism is used during Monte Carlo simulation, but more legible than simply passing the attribute identifier (e.g. 3).

A similar set of methods to `FGetAtt()`, etc... are defined on `Sim`, called `FFetchAtt()`, etc.. These are described in Section K.3 below, but they only differ from the `Editable::XGetAtt` methods in two respects. Firstly, because they are defined on `Sim`, they can "fetch" the attribute value using the currently live derivation route (whereas the `Editable::XGetAtt` methods can only return point values or a reserved value to signify UNKNOWN). Secondly, the `Sim::XFetchAtt` methods take an optional sequence of link identifiers as additional arguments - thus they can follow a link path to fetch an attribute value.

This concludes the overview of how object persistence and run-time schema querying are provided by classes `Editable`, `Model` and `ClassInfo`.

K.3 Evaluation Algorithm

K.3.1 Analysis

The previous section described the design of a `Model`, consisting of a network of `Editable` objects. In this section the provision of multiple derivation routes is described, (including uncertain values) and the mechanism for Monte Carlo simulation of such a model is presented. Some of the main classes involved in evaluation of the risk model are shown in Table K-2.

Class Name	Description
`Sim`	An object which may have multiple, possibly uncertain, derivation routes for each attribute value and whose attribute values can be evaluated by Monte Carlo simulation.
`SModel`	A network of `Editable` objects which may include objects of class `Sim` and `IAC`. The network may be evaluated by simulation.
`IAO`	Is-an-alternative-of relationship. An uncertain link between one source `Sim` object and one or more possible destination `Editable/Sim` objects. The destination object selected by the relationship generally varies during simulation.
`RouteData`	A description of all the derivation routes which are defined for the value of a `Sim` object attribute. Includes an indicative point value for the attribute (expected value or most likely value), and an indication of the currently "live" route (the most preferred route for which all necessary input information is available).
`Route`	A single derivation route for the value of a `Sim` object attribute.
`MethodRoute`	A derivation route which involves firing a method.
`AttRoute`	A derivation route which involves fetching an attribute value from elsewhere in the risk model.
`LinkDef`	Description of a link path which must be followed to reach one `Editable` object from another. Takes the form: <link1>[<card1>].<link2>[<card2>] ... <linkN>[<cardN>]
`SimData`	Data generated and used during a simulation for an attribute value or a link value.
`IAOSimData`	Data generated and used during simulation for a link value.
`AttSimData`	Data generated and used during simulation for an attribute value. Includes a single numeric sample value generated during each run of a simulation for an attribute of a `Sim` object (may be of type BOOL, FLOAT or INTEGER).
`Output`	An attribute value chosen by the user to be an output from the simulation. An output includes any goal values and goal probabilities which the user has defined and, optionally, an audit trail which is a depiction of the attribute derivation network for the output. Includes an instance of `UTarget`..
`UTarget`	An "uncertain target", an uncertain value which is constructed by repeatedly sending sample values to it during a Monte Carlo simulation - hence a "target". Stores a set of sample values. Can provide histogram bar heights and statistics to describe itself. After a simulation, an `Output` will contain a `UTarget`.
`UFloatTarget`	A `UTarget` which stores floating point sample values.
`UIntegerTarget`	A `UTarget` which stores integer sample values (also used for BOOLEAN samples).
`UncertainValue`	An `Editable` object representing a probability distribution with its parameters.
`PDF`	A probability distribution - a `PDF` object can be used to generate pseudo-random numbers with the desired distribution. Prior to a simulation, each `UncertainValue` taking part must create an instance of `PDF`.
`AuditParameters`	Parameters which determine the form of the audit trail - includes minimum class to display (only objects of this class and derived classes will be included in the trail) and maximum depth of indentation to be displayed.
`AuditTrail`	The audit trail, a string representation of the attribute derivation network for a particular attribute value.

Table K-2: Excerpt from data dictionary showing classes involved in evaluation of the risk model

K.3.1.1 Fusion Object Model

The classes `Sim` and `SModel` ("Sim Model") are specialisations of `Editable` and `Model` which additionally provide multiple derivation routes and the Monte Carlo engine. Figure K-15 shows that a `Sim` object contains both an instance of `RouteData` for each attribute value (which describes the multiple derivation routes) and an instance of `AttSimData` ("Attribute Simulation Data", which includes a single numerical sample value, generated during Monte Carlo simulation). The `SModel` contains zero or more `Outputs`, which represent the output attributes selected by the user. In an `SModel`, only some of the `RouteData` objects will vary their sample values during a Monte Carlo simulation, and hence need to be re-sampled by the `SModel` - this is shown by the "non-constant routes" relationship. Similarly, not all of the `IAO` relationships will vary their selected object during a simulation, and those which will vary are shown related to `SModel` by "non-constant IAOs".

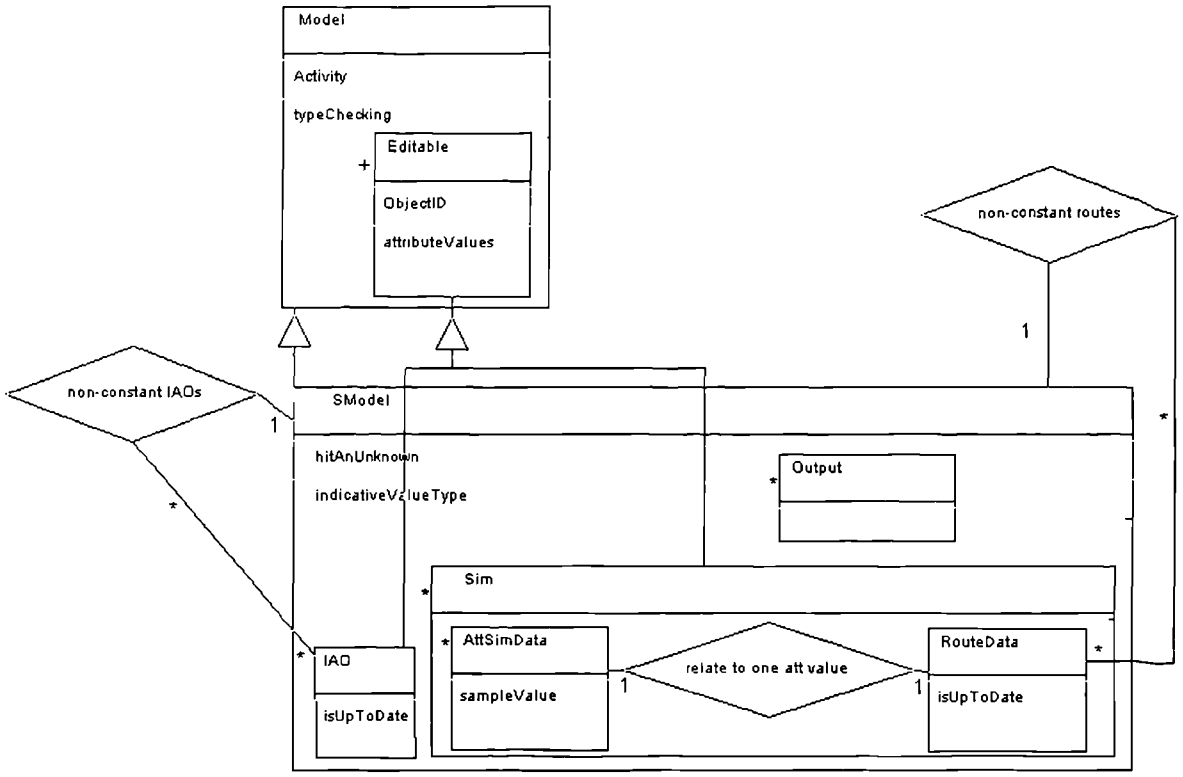


Figure K-15: Object model for SModel ("Sim Model")

A new attribute `Model:typeChecking` is shown - this is a Boolean flag which enables consistency checking against the schema to be switched off to save time during a simulation. The attribute `SModel::hitAnUnknown` is described below, and `SModel::indicativeValueType` flags whether the expected value or the most likely value should be calculated and displayed as the indicative point value for attributes in the model. Most of the other attributes are hidden in Figure K-15, but the `isUpToDate` flags for `IAO` and `RouteData` are shown. These are used during a simulation to ensure that each random variable taking part in the simulation is sampled exactly once per simulation run.

The simulation data classes derived from `SimData` are shown in Figure K-16. There are two types of simulation data - that which relates to attribute values (`AttSimData` as mentioned above) and that which relates to uncertain link values (`LinkSimData`). Both types may or may not vary during a simulation (indicated by the `isConstant` Boolean attribute) and both types may simply take a point value. It is useful to record these two attributes as part of the simulation data so that they can be quickly accessed during simulation. For `AttSimData` the sample value generated during a single run is a numerical value, and for `LinkSimData` the sample link value is a pointer to an Editable object.

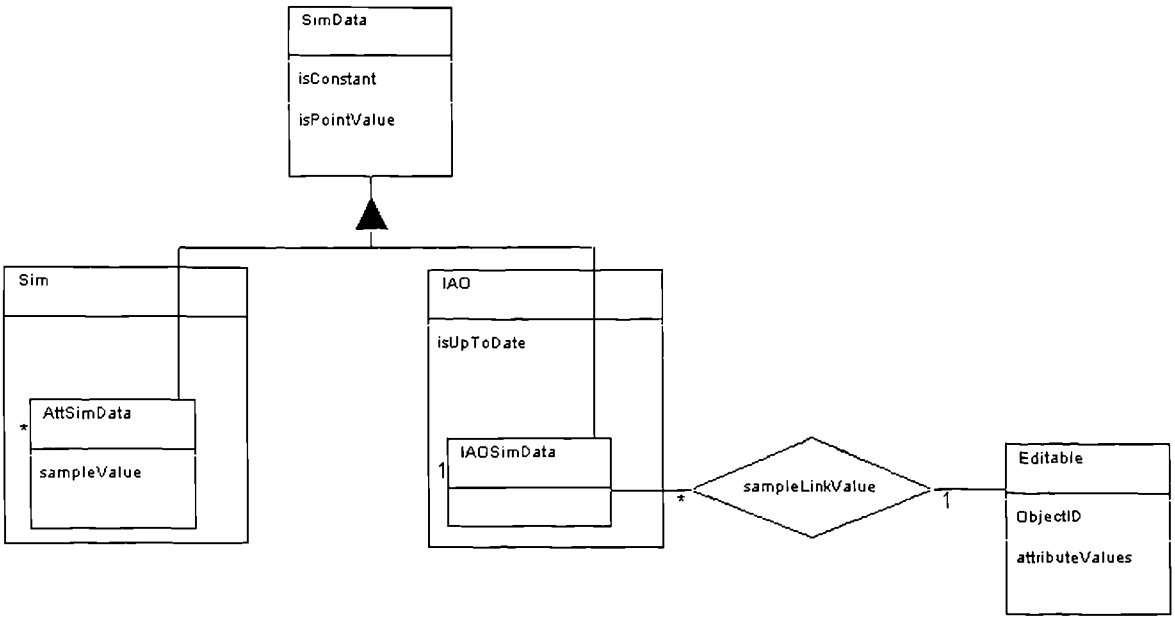


Figure K-16: Object model for SimData

Figure K-17 shows that the route data stored for each attribute value of a `Sim` object consists of zero or more derivation routes, each of which may be a method route, an attribute route or an uncertain value (if a point value is defined for an attribute value then this is stored in `Editable`, not `Sim`). The route data also includes a pointer to the currently live route, an indicative point value and a flag to indicate whether or not the information in `RouteData` is up to date. When the live routes and indicative values are being determined, this flag is used to indicate whether their values are up to date. During a simulation, the same flag is also used to indicate whether the corresponding `AttSimData` is up to date (i.e. has been sampled during this simulation run). Both the method route and the attribute route contain a link definition - this specifies the link path which must be followed to reach the `Editable` object on which the method should be fired (or from which the attribute value should be copied).

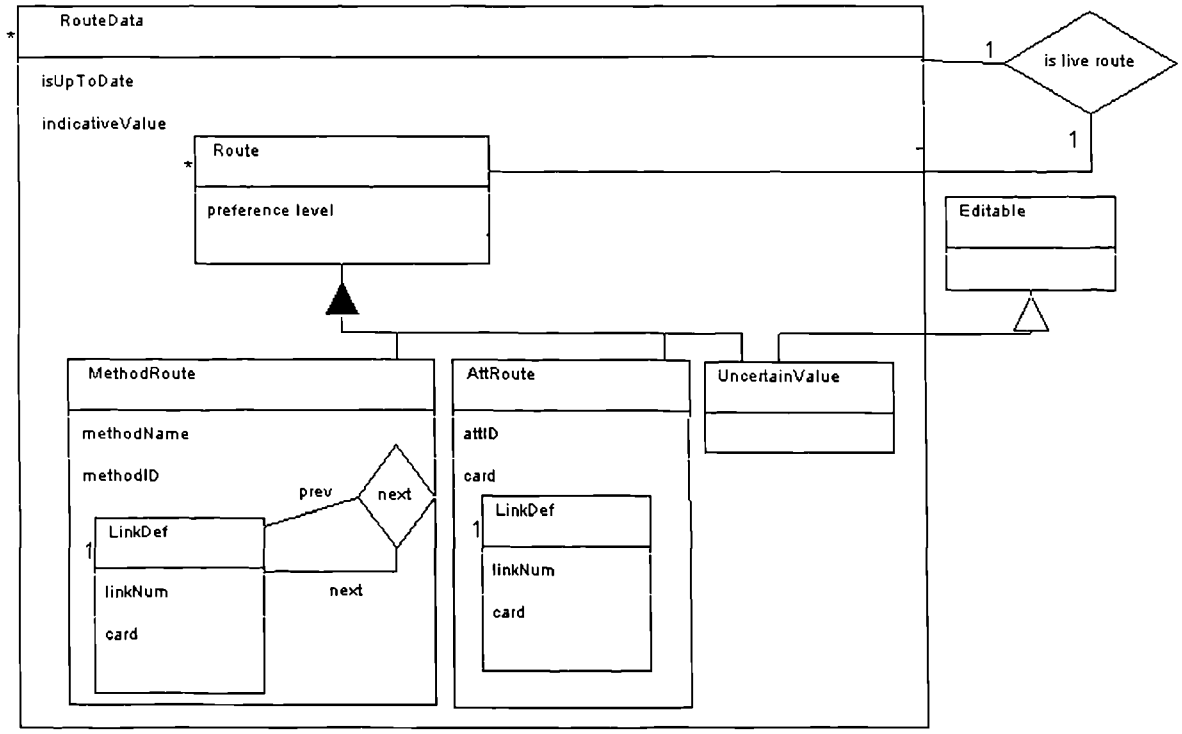


Figure K-17: Object Model for RouteData (one instance is stored per attribute value)

Figure K-18 shows that an `Output` contains one of two types of `UTarget`, used to store either floating point or integer sample values - the set of sample values generated for a Boolean attribute is stored as a `UIntegerTarget`. The samples values are stored both in their original sampled order, as

`storedValues`, and also sorted into ascending order, as `sortedValues`. This enables percentiles and histograms for the cumulative PDF to be generated quickly from existing samples. `UTarget.rankValues` contains the rank of each sample value, stored in the same order as `UFloatTarget.sampleValues` or `UIntegerTarget.sampleValues` - the ranks are used to calculate the correlation coefficients.

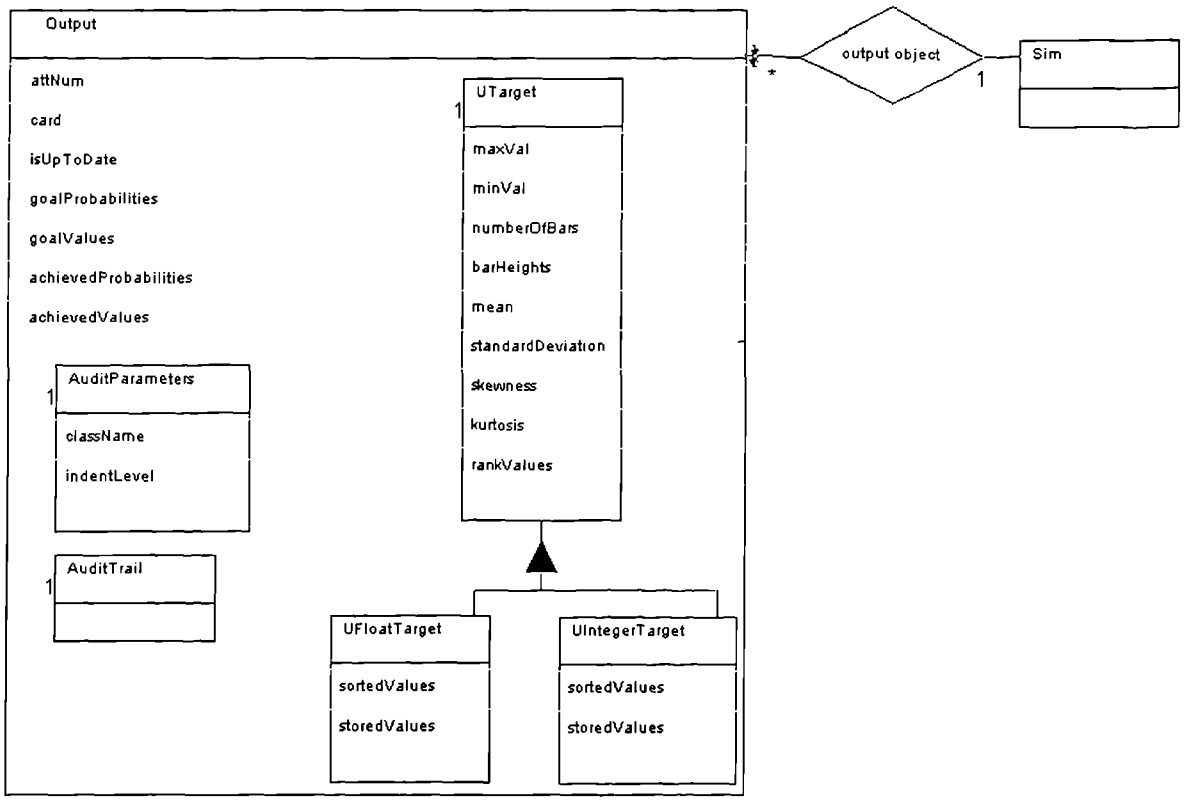


Figure K-18: Object Model for Output

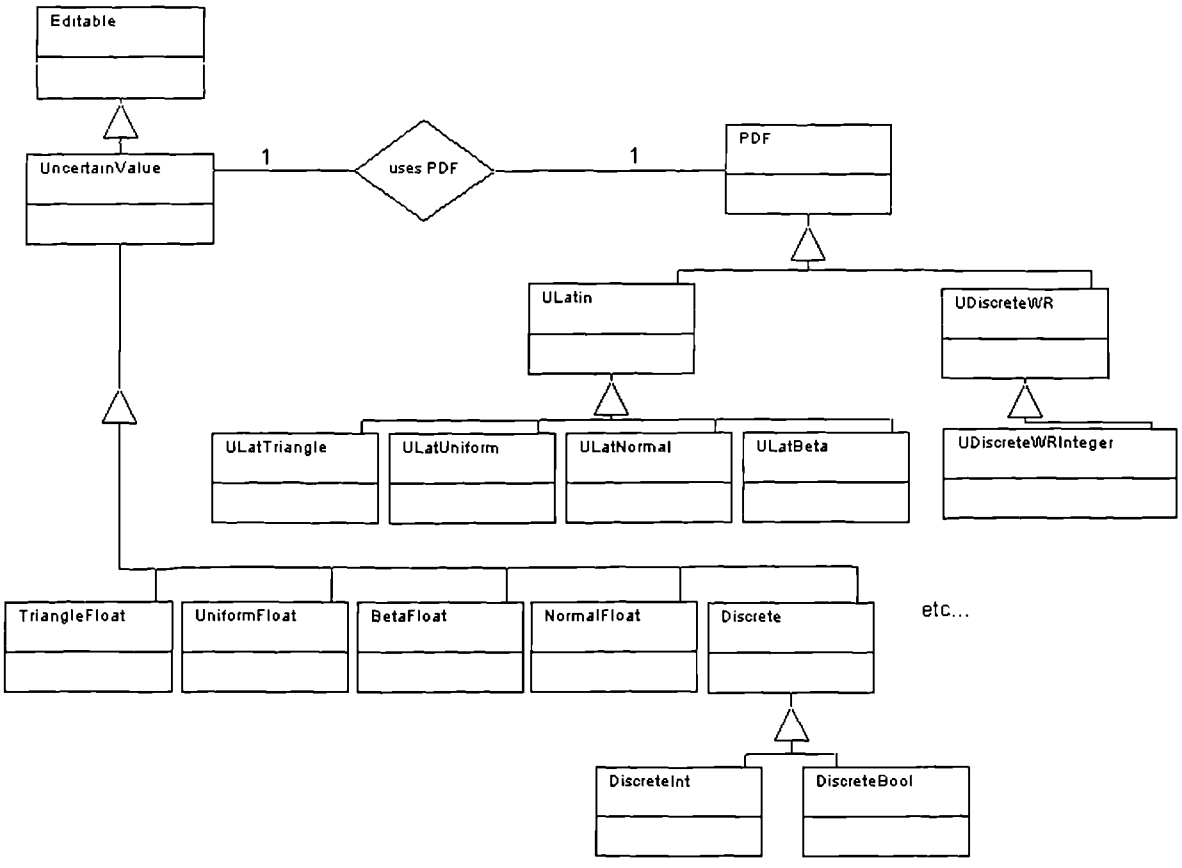


Figure K-19: Partial object model shows some of the UncertainValue and PDF classes

Each instance of `UncertainValue` involved in a Monte Carlo simulation creates an instance of `PDF` which is used to generate the pseudo-random numbers required during the simulation (Figure K-19). Thus, there are two classes corresponding to each probability distribution - this approach has the advantage of decoupling the sampling method from the definition of the probability distribution and its parameters, and also limits the memory consumed to that which is required to simulate the portion of the risk model necessary for the specified output attribute. Considering a beta-distribution for example, an instance of `BetaFloat` will create an instance of `ULatBeta` prior to a simulation. All classes derived from `ULatin` use Latin hypercube sampling, and all classes derived from `UDiscreteWR` use sampling without replacement of discrete values.

K.3.1.2 Fusion Interface Model

The system operations involved in performing a Monte Carlo simulation are shown in Figure K-20 and their operation models are shown below.

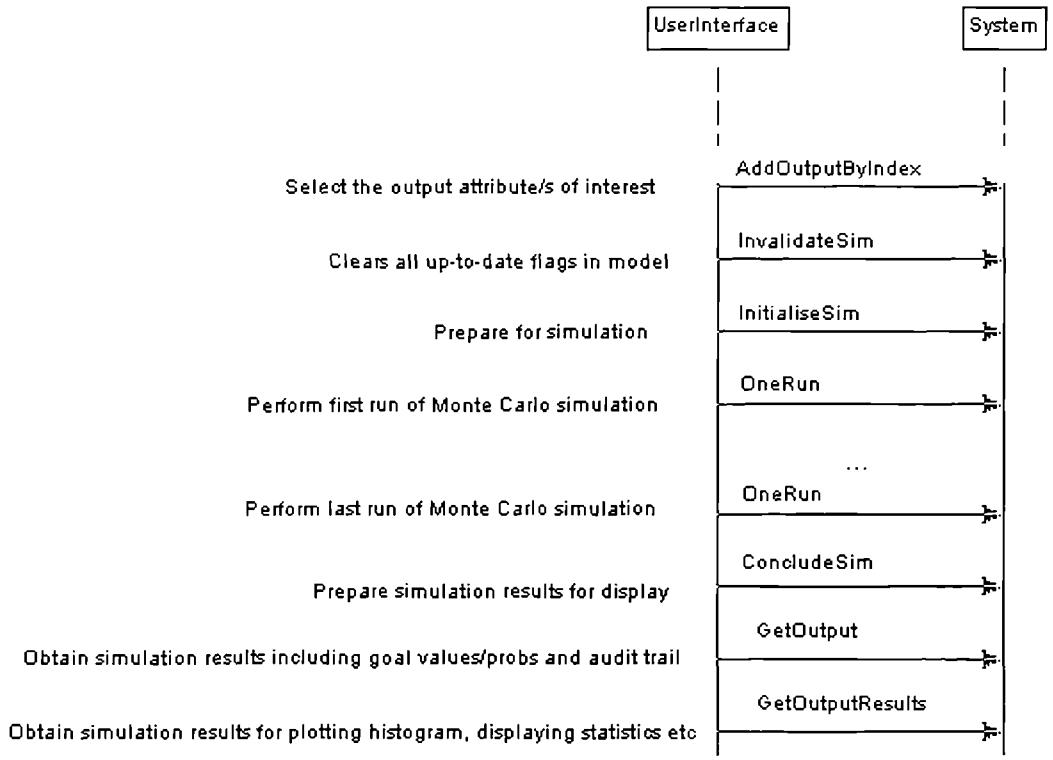


Figure K-20: Example time-line scenario for Monte Carlo simulation

```
Operation : AddOutputByIndex

Supplied : smodel_object : SModel ,
           sim : Sim ,
           i : INDEX

Description : " Adds attribute value with index i of object sim to the
list of output attributes for smodel_object. (The probability
distribution for each output attribute will be evaluated during a Monte
Carlo simulation of smodel_object). "

Changes : smodel_object : SModel ,
          col outputs : BoundList(Output) with " output belongs to
smodel_object " ,
          new output : Output

Assumes : " An attribute value with index i exists in object sim "

Result : " A new output has been created and added to smodel_object "
```

```
Operation : InvalidateSim

Supplied : smodel_object : SModel

Description : " Invalidates the results of a previous simulation. "

Changes : smodel_object : SModel ,
          col outputs : BoundList(Output) with " output belongs to
smodel_object "

Result : " output.isUpToDateFlag has been assigned a value of FALSE for
all outputs which belong to smodel_object. "
```

```
Operation : InitialiseSim

Supplied : smodel_object : SModel ,
           numberIts : INTEGER ,
           seed : INTEGER

Description : " Called immediately prior to a simulation, this operation
prepares smodel_object for a simulation. If there is no indicative value
for one of the outputs in smodel_object (i.e. it is Unknown), or the
simulation is already up-to-date, then that output will be omitted from
the simulation - but smodel_object is prepared for simulation of all
other outputs. Builds the collections nonConstantIAOs, nonConstantRoutes
and pdfs whuch are needed during simulation and creates the instances of
SimData which are also required. Initialises all UncertainValue objects
which will be taking part in the simulation. Seeds the random number
generator. Switches type checking off in smodel_object, for speed during
simulation, and sets its activity to be SIMULATING. "

Changes : static_pdf : PDF ,
          smodel_object : SModel . typeChecking : BOOL ,
          smodel_object : SModel . activity : Activity ,
          new col pdfs : List(UncertainValue) ,
          new col nonConstantIAOs : List(IAO) ,
          new col nonConstantRoutes : List(RouteData) ,
          new attSimData : AttSimData ,
          new linkSimData : LinkSimData ,
          new target : UTarget ,
          col outputs : BoundList(Output) with " output belongs to
smodel_object, has known indicative point value and output.isUpToDate ==
FALSE "

Result : " A new UlFloatTarget or UIntegerTarget, as appropriate, has
been created in each output taking part in the simulation. The lists
nonConstantIAOs, nonConstantRoutes and pdfs
have been created and filled. Instances of AttSimData have been created
for each value of each attribute of every Sim object taking part in the
simulation. An instance of LinkSimData has been created in each IAO
taking part. All UncertainValue objects taking part have been prepared to
generate numberIts pseudo-random values. The random number
generator in static_pdf has been seeded with a value of seed.
smodel_object.type_checking == FALSE.
smodel_object.activity == SIMULATING. "
```

```
Operation : OneRun

Supplied : smodel_object : SModel

Description : " Performs one run of a simulation. Initially, clears the
up-to-date flags in nonConstantRoutes and nonConstantIAOs. Calculates one
value for each output attribute (of those which are not Unknown and whose
samples were not up-to-date at the start of the simulation). Stores the
result in the corresponding UTarget object for each output. "

Changes : col outputs : BoundLis(Output) with " output belongs to
smodel_object, has known indicative point value and output.isUpToDate ==
FALSE " ,
          col nonConstantRoutes : List(RouteData) ,
          col nonConstantIAOs : List(IAO)

Assumes : " smodel_object.activity == SIMULATION "

Result : " IAO::isUpToDate and RouteData::isUpToDate have been set TRUE
for each element of nonConstantIAOs and nonConstantRoutes which has been
evaluated during this run. One additional sample value has been stored to
the UTarget object for each Output involved in the simulation.  "
```

```
Operation : ConcludeSim

Supplied : smodel_object : SModel

Description : " Concludes the Monte Carlo simulation. Deletes objects
which were created temporarily for use during the simulation,releasing
memory. Calculates histogram bar heights and statistics for the
sample values created during the simulation. Switches type checking on
for smodel_object and sets its activity to be browsing. "

Changes : col outputs : BoundList(Output) with " output belongs to
smodel_object " ,
          col pdfs : List(UncertainValue) ,
          col nonConstantIAOs : List(IAO) ,
          col nonConstantRoutes : List(RouteData) ,
          attSimData : AttSimData ,
          linkSimData : LinkSimData ,
          smodel_object : SModel . activity : Activity ,
          smodel_object : SModel . typeChecking : BOOL

Result : " All instances of AttSImData and LinkSimData which were present
in the risk model have been deleted. The collections pdfs,
nonConstantIAOs and nonConstantRoutes have been deleted.  The UTarget
objects in the Outputs in the risk model have histogram bar heights and
statistcs describing the sampled values available.
smodel_object.activity == BROWSING.
smodel_object.typeChecking == TRUE. "
```

```
Operation : GetOutput

Supplied : smodel_object : SModel ,
           i : INDEX

Returns : Output

Description : " Returns the i'th output from smodel_object. "

Reads : smodel_object : SModel ,
        col outputs : BoundList(Output) with " outputs belong to
smodel_object "

Assumes : " There are at least i + 1 outputs defined for smodel_object. "

Result : " Unchanged "
```

```
Operation : GetOutputResults

Supplied : smodel_object : SModel ,
           i : INDEX

Returns : UTarget

Description : " Returns the uncertain target object for the i'th output
from smodel_object. "

Reads : smodel_object : SModel ,
        output : Output with " output is i'th output in smodel_object "

Assumes : " There are at least i + 1 outputs defined for smodel_object. "

Result : " Unchanged "
```

K.3.2 Design

Before analysing how the system operations above are performed, the significance of the `SModel::activity` flag must be described, and this is related to a set of methods named `FFetchAtt()` (for "floating point fetch" attribute), `IFetchAtt()` (for integer-typed attributes), and `BFetchAtt()` (for Boolean) which are defined on `Sim`. These are similar to the `FGetAtt()`, `IGetAtt()` and `BGetAtt()` methods defined on `Editable` and described in Section K.3 above, except that they use the information stored in `RouteData` for the requested attribute value to attempt to fetch the requested value from the currently live route, and can also (optionally) follow one or more links to obtain the information.

```
Method : Sim.FFetchAtt

Supplied : A : ATTNUM ,
           C : CARD ,
           links : List(LINKNUM) ,
           link_cards : List(CARD)

Returns : FLOAT

Description : " Fetches a value for attribute A, cardinality C of the Sim
object which can be reached by following the links stored in links and
link_cards. Follows the currently live derivation route to obtain the
value. "

Reads : routeData : RouteData with " routeData describes specified
attribute value "

Changes : smodel : SModel with " this Sim object belongs to smodel "

Assumes : smodel.activity == SIMULATING

Result : " If an Unknown attribute value or link value was encountered at
any stage during the method execution then smodel.hitAnUnknown has been
assigned a value of TRUE. "
```

These "XFetch" methods are one of the key elements of the evaluation mechanism for RiTo. Methods defined on a `Sim` derived class (for example `Assembly`) will obtain their input data by invoking the XFetch methods. Suppose the live route for a requested attribute value is a `MethodRoute` - for example the live route for `part_22.piece_cost` might be the method `Assembly::piece_cost_fn()`. When `part_22.FFetchAtt(ATT_piece_cost)` is invoked, the `FFetchAtt` implementation will invoke `piece_cost_fn()`. This method body will contain calls to XFetch methods as it obtains its necessary input data, for example:

```
FFetchAtt(ATT_material_cost, 0, LNK_manufacturing_process),
```

and so this invocation may then lead (directly or indirectly) to other calls to the XFetch method, and so on. Thus eventually all the inputs which are needed (directly or indirectly) to evaluate the piece cost of part_22 are obtained recursively. If the live route is an `UncertainValue`, then the XFetch methods return the next random number generated from this value. If the live route is an `AttRoute`, then XFetch is invoked on the

specified attribute value. And if the live route is a point value, then the point value is simply returned. The XFetch methods also deal with any uncertain link values (IAOs) which are encountered. If there were several alternative manufacturing processes for part_22, above for example, then `FFetchAtt()` would obtain the next random sample for `selection` from the `IAO` and then fetch the material cost from this selected manufacturing process. The implementation is slightly more complex when there may be several nested links to follow, each of which may have alternatives, but the principle remains the same.

The description above outlines the role of the XFetch methods during a Monte Carlo simulation - i.e. when `SModel::activity` has a value of SIMULATING. However, since their invocations are hard-coded into the method bodies for user-defined `Sim` classes, the same XFetch methods must also be used when preparing for simulation and when calculating the live routes and indicative values for the risk model, and this is where the `SModel::activity` flag becomes important. The behaviour of the XFetch methods varies depending upon the value of `SModel::activity`. When obtaining indicative values, the behaviour is much as described above except that instead of taking a random sample from each `IAO` and `UncertainValue` encountered, the expected value (or most likely value, according to `indicativeValueType`) is used.

When preparing for a simulation, three lists which will be needed during the simulation are constructed by the XFetch methods - a list of `UncertainValues` which will participate in the simulation, a list of `RouteData` objects for the attribute values which will vary during the simulation (non-constant routes) and a list of the `IAO` objects whose selected objects will vary during the simulation (non-constant `IAOs`). Thus the behaviour is similar to that during a simulation, but instead of taking a sample from each `Uncertainvalue` and `IAO` encountered, the object is added to the appropriate list. Also, new instances of `AttSimData` and `LinkSimData` are created for each `Sim` and `IAO` object encountered, respectively.

When calculating the live routes, each route must be tested, beginning with the most preferred route, until one is found which is able to return a value. The flag `SModel::hitAnUnknown` is used to indicate when an UNKNOWN link or attribute value has been encountered. The XFetch methods return a special reserved value when the requested attribute value is unknown (the largest expressible value for the type),and the user-defined method bodies in `Sim` classes test the `hitAnUnknown` flag before performing arithmetic operations on the returned value - thus avoiding floating point errors.

Having outlined the role of `SModel::activity` and the XFetch methods in the evaluation of a risk model, we now return to a description of each the system operations from Figure K-20, which are involved in performing a Monte Carlo simulation (the system operation models were given in Section 3.1.2 above). `AddOutputByIndex` and `InvalidateSim` are self-explanatory from their operation models. The implementation of `InitialiseSim` is considerably more complex. Some interactions have been omitted from the object interaction graph shown in Figure K-21 in the interest of simplicity, and `FFetchAtt` is shown as an example of an XFetch method.

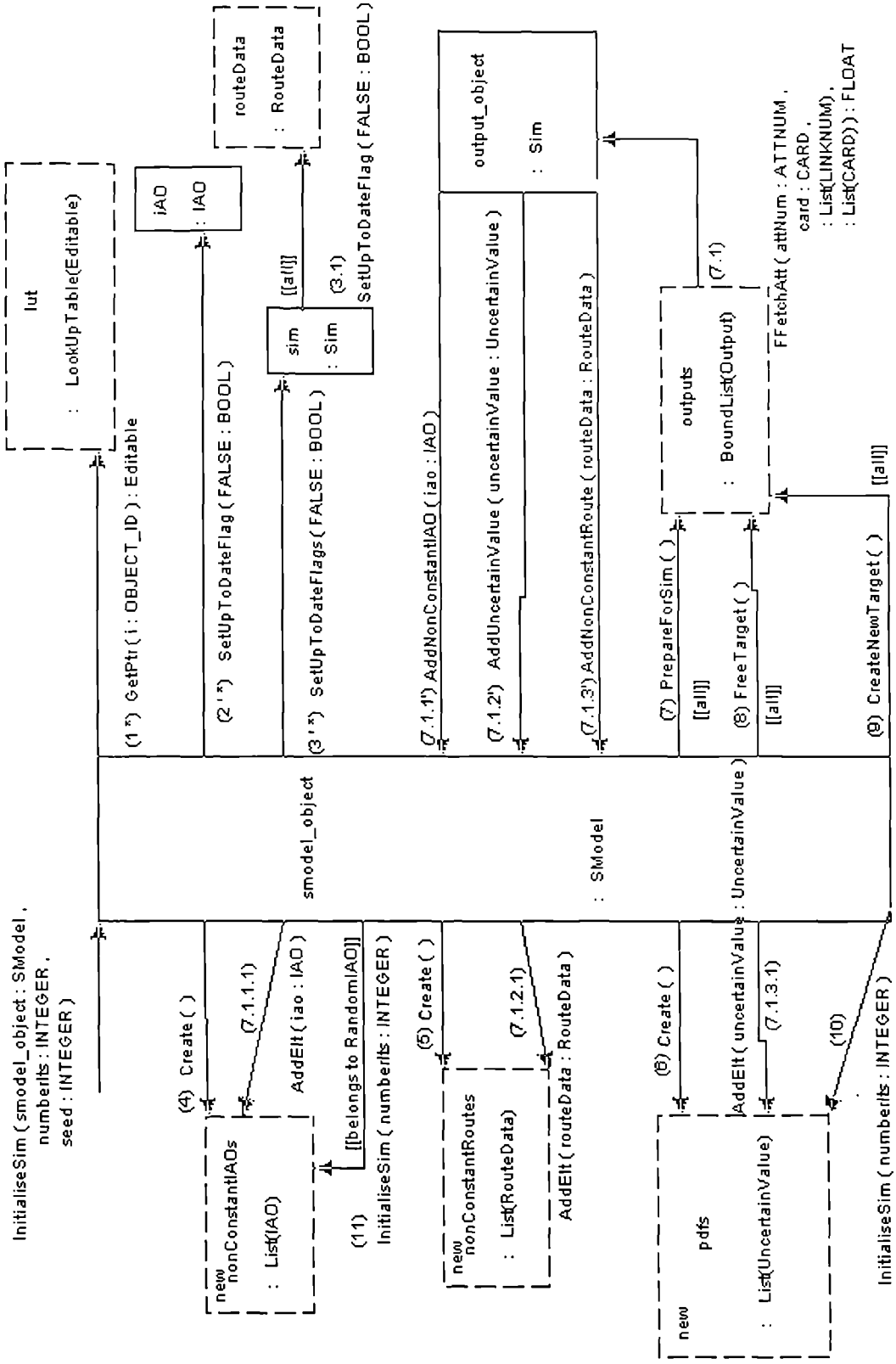


Figure K-21: Partial OIG for system operation InitialiseSim

The description of `InitialiseSim` is given below:

```
operation SModel::InitialiseSim(INTEGER numberIts, INTEGER seed)
/* Prepares this SModel for a simulation.
*/

hitAnUnknown = FALSE

FOR i = 0 TO MAX_NUM_OBJECTS
 (1) ed = Editable object with ID i from risk model.
 (2) IF ed belongs to class IAO THEN set editable.upToDate = FALSE
     IF ed belongs to class Sim THEN
     (3) FOR routeData = each RouteData object in ed
          (3.1) set routeData.upToDate = FALSE.
         NEXT routeData
     ENDIF
NEXT i

activity = PREPARING_SIMULATION

typeChecking = TRUE

(4) Create new list nonConstantIAOs
(5) Create new list nonConstantRoutes
(6) Create new list pdfs

FOR output = each output which has an indicative value and is not
upToDate.
       (7) CALL FFetchAtt(output.attNum, output.card) on the
           output_object specified in output.

           This call adds elements to:
           (7.1.1) Model::nonConstantIAOList
           (7.1.2) Model::PDFList
           (7.1.3) Model::nonConstantRouteList
           as they are encountered.

           The FFetchAtt also creates AttSimData for every
           att value of each Sim object encountered. Similarly creates a
           single LinkSimData in each IAO encountered. (Not shown on
           Object Interaction Graph).

       (8) CALL output.FreeTarget. This deletes any existing target for
           the output and invalidates the stored values for
           achievedProbabilities and achievedValues.

       (9) Create a new target for the output.

NEXT output

activity = SIMULATING

Assign a seed value of seed to the random number generator in PDF (not
shown on OIG).

(10) CALL InitialiseSim(numberIts) on each element of pdfs.

(11) CALL InitialiseSim(numberIts) on the random variable for each
element of nonConstantIAOs which belongs to class RandomIAO.

typeChecking = FALSE
```

```
method UncertainValue::InitialiseSim(INTEGER numberIts)
/*
Creates a new PDF of the appropriate class and passes it the attributes
of this Editable object.
*/
```

The method `UncertainValue::InitialiseSim` (above) is implemented in derived classes - for example Figure K-22 shows the initialisation of a uniform distribution - this involves the creation of a new

ULatUniform object, capable of generating pseudo-random numbers using the Latin hypercube sampling method (described in Chapter 9, Section 9.1). The classes used for generating pseudo-random numbers, such as ULatUniform, are described in Section 3.3, below.

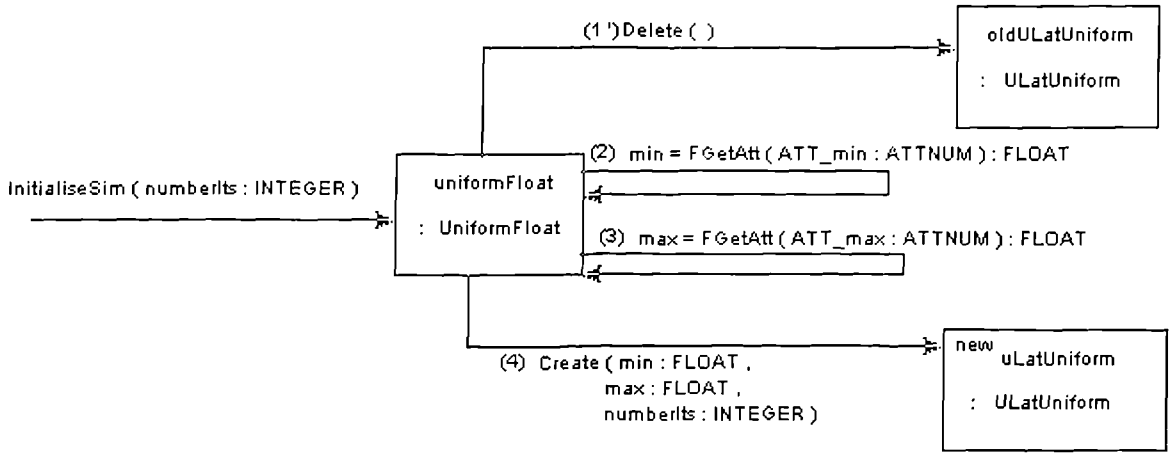


Figure K-22: Example of an OIG for an UncertainValue::InitialiseSim

Having prepared for the simulation, operation OneRun is then invoked numberIts times:

```
method SModel::OneRun()
/* Performs one run of a simulation.
*/
Clear the upToDate flag on each element of nonConstantRoutes.

Clear the upToDate flag on each element of nonConstantIAOs.

FOR output = each output which has an indicative value and is not
upToDate.

  CALL FFetchAtt(output.attNum, output.card) on the output_object
  specified in output, and send the result to the UTarget stored in
  output.

NEXT output
```

Finally, ConcludeSim is called (see system operation model above, and method description below). The calling application then invokes GetOutput or GetOutputResults (see system operation models above) to obtain the necessary information for display to the user.

```
method SModel::ConcludeSim()
/*
 Concludes simulation.
*/

FOR i   0 TO MAX_NUM_OBJECTS
 ed = Editable object with ID i from risk model.
 IF ed belongs to class IAO THEN delete the IAOSimData stored in ed
 IF ed belongs to class Sim THEN
  delete all instances of AttSimData stored in ed
 ENDIF
NEXT i

Delete nonConstantIAOList.

Delete nonConstantRouteList.

Delete pdfs.

FOR output = each output which has an indicative value and is not
upToDate.

 CALL Generate on its UTarget - this causes the sample values to be
 sorted and ranked and histogram bar heights and statistics to be
 calculated.

 output.isUpTodate = TRUE

NEXT output

typeChecking = TRUE

activity = BROWSING
```

To conclude this section, visibility graphs for the main classes described above are shown in Figure K-23 to Figure K-25.

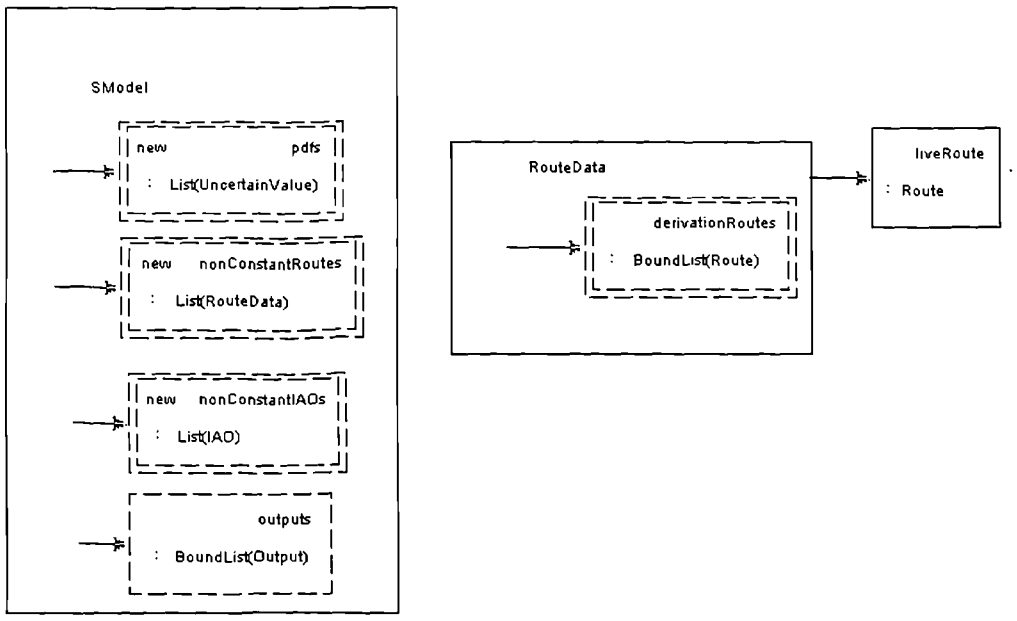


Figure K-23: Visibility graphs for SModel and RouteData

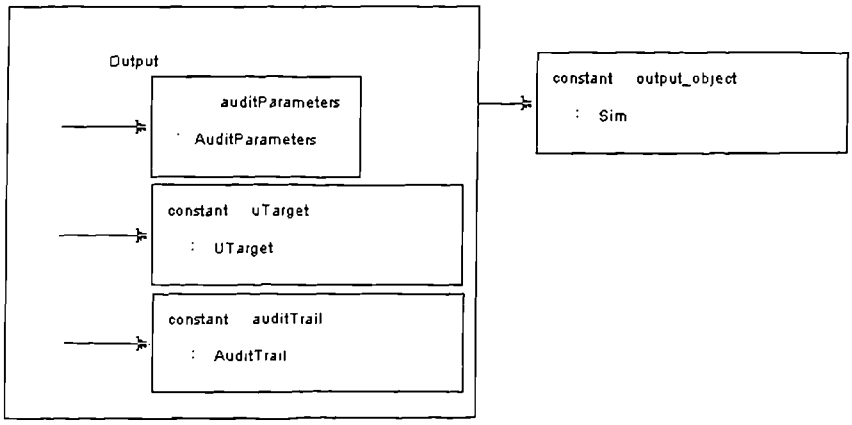


Figure K-24: Visibility graph for Output

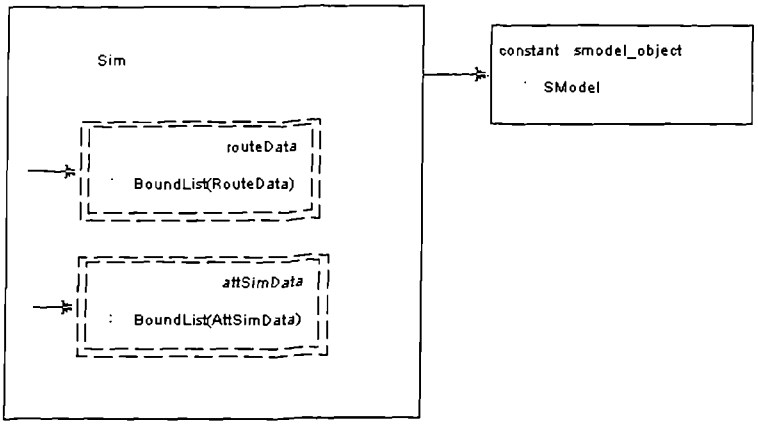


Figure K-25: Visibility graph for Sim

K.3.3 Random Number Generation Classes

The main random number generation classes are `PDF`, `ULatin` and `UDiscreteWR`, shown in Figure K-26 with one example of a continuous distribution function (`ULatUniform`) and one example of a discrete distribution (`UDiscreteWRInteger`).

All random number generating classes are derived from PDF - this class provides the pseudo-random number generator using a shuffling table and three linear congruential generators (LCGs) which was described in Section 9.1.2 of Chapter 9. The generator is seeded using method `Seed`, and samples are obtained by invoking `UniformRand` - this method implements the algorithm given in Section 9.1.2. The constants `Max1..Max3`, `Ai..A3` and `Ci..C3` are static data members, stored only once for the whole class. The data members `X1`, `X2` and `X3` are used by the three LCGs and are also static. It was found that the time taken to generate random numbers was considerably longer when the `UniformRand()` method was used than when the system-supplied `rand()` function was used. For this reason, a compiler switch was included to allow the programmer to choose between the two random number generators at compile-time.

Most of the functionality required for Latin hypercube simulation is provided in `ULatin` - the derived classes, such as `ULatUniform`, need only provide data members for the distribution parameters, a constructor (the `Create` method shown in Figure K-22) and also a method which calculates the inverse cumulative distribution function (CDF). The inverse CDF must be evaluated for each pseudo-random number which is generated and thus in most of the `ULatin` classes, this is made more efficient by also providing data members which are intermediate variables, calculated from the distribution parameters. For example, in a uniform distribution, the inverse CDF of x is given by $min + (range * x)$, where $range = max - min$.

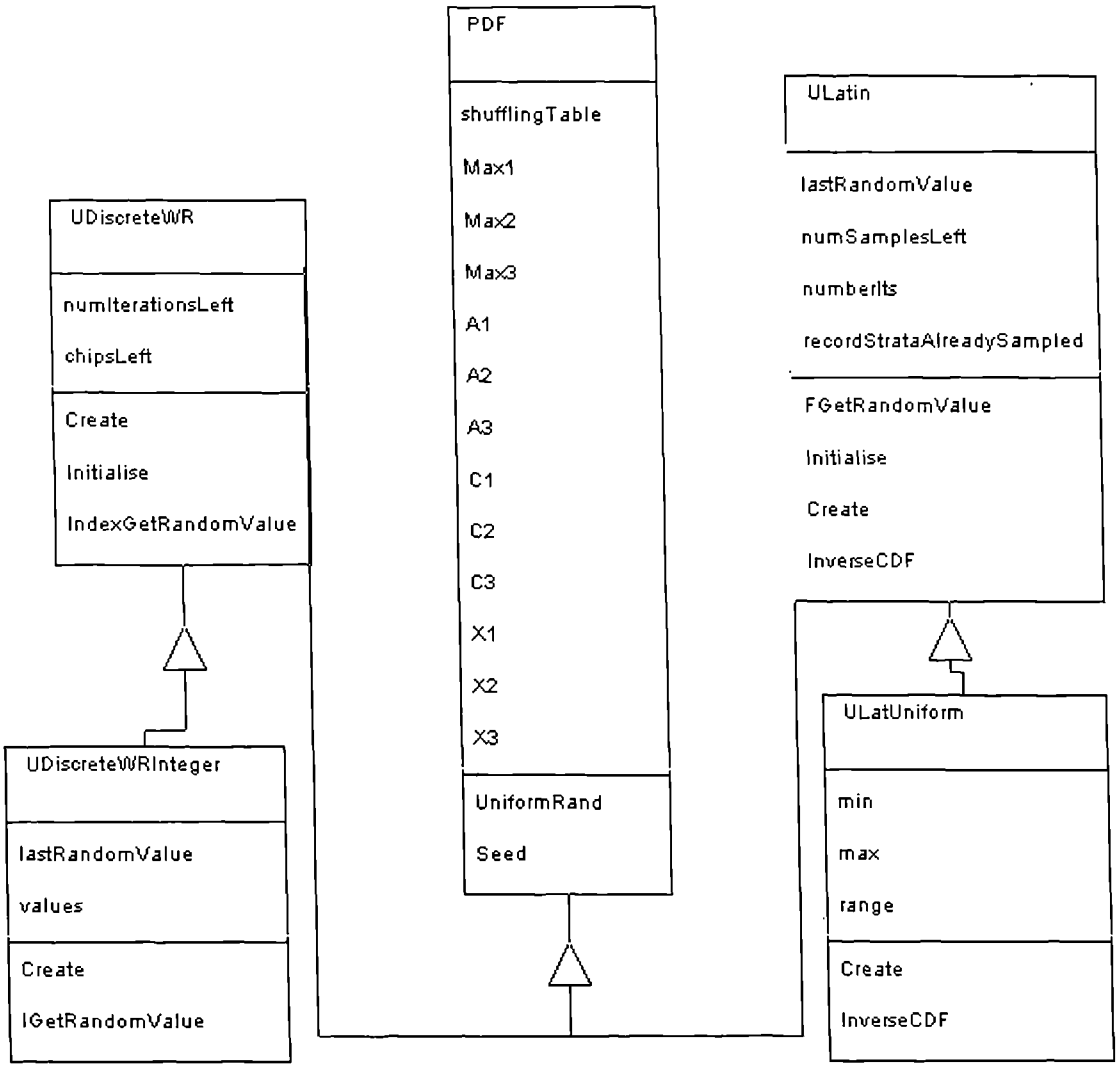


Figure K-26: Inheritance graph illustrating main random number generation classes

The class definitions for `ULatin` and `ULatUniform` are given below:

```
class ULatin isa PDF
 method FGetRandomValue (  ) : FLOAT
 method Initialise ( numberIts : INTEGER )
 method Create (  ) : ULatin
 method InverseCDF ( cumulativeProb : FLOAT ) : FLOAT
 attribute variable lastRandomValue  : FLOAT
 attribute variable numSamplesLeft  : INTEGER
 attribute variable numberIts  : INTEGER
 attribute variable recordStrataAlreadySampled  : Array(BIT)
endclass
```

```
class ULatUniform isa ULatin
 method Create ( min : FLOAT ,
                 max : FLOAT ,
                 numberIts : INTEGER )
 attribute variable min  : FLOAT
 attribute variable max  : FLOAT
 attribute variable range  : FLOAT
 method InverseCDF ( cumulativeProb : FLOAT ) : FLOAT
enclass
```

Random values are obtained by invoking `ULatin::FGetRandomValue`, which in turn invokes `InverseCDF`, according to the algorithm given in Section 9.1.3 of Chapter 9; the method `ULatin::InverseCDF` is only implemented in derived classes.

A similar system has been used to provide integer sampling without replacement; the main functionality is provided by `UDiscreteWR`, with derived classes (one for each type) storing the array of discrete values and providing a method to return a random sample of the appropriate type - for example `IGetRandomValue()` which returns an INTEGER value, is provided by `UDiscreteWRInteger` as illustrated in Figure K-26 above.

This concludes the description of the evaluation algorithm used by RiTo.

K.4 Class-based Interface with DLL

A set of pure virtual classes are defined which provide an interface between RITO.EXE and SIMDLL.DLL (as described in Section 10.1 of Chapter 10). The main classes are listed in Table K-3. In addition to these classes, there are various simple "building block" classes which are defined in SIMDLL.DLL and used in both RITO.EXE and SIMDLL.DLL (often as method arguments) - for example `Str` (string class), `AuditPars` (audit trail parameters), `HistPars` (histogram parameters), `DLlistStr` (doubly-linked list of strings) etc. The building block classes are not described further here.

Class Name	Description
`SDRoot`	Classes derived directly or indirectly from SDRoot can be created/destroyed using global functions in SIMDLL.DLL.
`SModel_SD`	The risk model (see `SModel` in Table K-2).
`Editable_SD`	Object capable of persistence and responding to schema queries (see `Editable` in Table K-1).
`ClassInfo_SD`	The meta-class (see `ClassInfo` in Table K-1).
`Output_SD`	An output attribute (see `Output` in Table K-2).
`UTarget_SD`	A set of sample values generated during simulation (see `UTarget` in Table K-2).
`DLlistOutputPtr`	Doubly-linked list of pointers to `Output_SD` objects.
`DLlistTargPtr_SD`	Doubly-linked list of pointers to `UTarget_SD` objects.

Table K-3: The class-based interface between RITO.EXE and SIMDLL.DLL

Global functions named `SimDLLCreate()` and `SimDLLDelete()` are provided by SIMDLL.DLL, and the calling code can use these functions to directly create and destroy classes derived from `SD_Root` - of which there are only two, `SModel_SD` and `DLListTargPtr_SD`. Objects of all other classes are created and destroyed automatically by SIMDLL.DLL as a result of method invocations on other objects. For example, `Sim` objects are created by invoking a method `CreateNewObject` on an object of class `SModel_SD`.

The Fusion class definitions for the main interface classes are given below. The class definitions list the methods defined - each of these methods is implemented in a derived class. In some cases it was necessary to distinguish the name of the interface class method from the name of its implementation - in these cases `_SD` has been appended to the end of the name of the interface class method. Several methods which are strictly only required on `Sim` objects have been defined on `Editable_SD` - for example `Editable::GetIndValByIndex`, which returns the indicative value for an attribute of a `Sim` object, will simply return the point value on an `Editable` object. This avoids the need for a separate `Sim_SD` class, and `Editable_SD` is used in its place.

```
class SModel_SD isa
 method InvalidateSim          (  )
 method InitialiseSim          ( numberIts : INTEGER,
                                 seed : INTEGER )
 method RecalcAllLiveRoutes    (  )
 method OneRun                 ( )
 method ConcludeSim            (  )
 method GetOutputName          ( i : INDEX ) : Str
 method ClearAllUpToDateFlags  ( )
 method IsOutputSampleUpToDate( i : INDEX   : BOOL
 method InitializeWrite        ( activity : Activity )
 method ReadInModel            ( PathName : Str,
                                 TopObjectID : OBJECT_ID ):ErrorResultType
 method WriteOutModel          ( PathName : Str) : ErrorResultType
 method CycleIndValType        (  )
 method GetIndValType          (  ) : IndValType
 method GetHistDefaults        (  ) : HistPars
 method SetHistDefaults        ( histogramParameters : HistPars )
 method SetAllOutputsToHistDef(  )
 method AddOutputByIndex       ( sim : Editable_SD,
                                 i : INDEX)
 method SetSelectedProduct     ( productIndex : CARD ) : ErrorResultType
 method GetSelectedProduct     ( productIndex : CARD ) : Str
 method CalculateProdVols      (  ) : ErrorResultType
 method CheckProductModel      (  ) : ErrorResultType
 method GetProductNames        (  ) : DLlistStr
 method RemoveOutput           ( i : INDEX )
 method GetOutputResults_SD    ( i : INDEX  ) : UTarget_SD
 method GetOutput_SD           ( i : INDEX ) : Output_SD
 method GetTopObject_SD        (  ) : Editable_SD
 method GetNumOutputs          (  ) : INDEX
 method RenumberLUT            ( OBJECT_ID ) : OBJECT_ID
 method CreateNewObject_SD     ( className : Str,
                                 constructorArguments : Str,
                                 object_id : OBJECT_ID) : Editable_SD
 method GetPtr_SD              ( object_id : OBJECT_ID ) : Editable_SD
 method DeleteObject_SD        ( object_id : OBJECT_ID )
endclass
```

```
class Editable_SD isa
 method GetDerivationRoutes    ( A : ATTNUM,
                                 C : CARD ) : DLlistStr
 method IsKindOf               ( parent : ClassInfo_SD ) : BOOL
 method GetLinkClass_SD        ( L : LINKNUM ) : ClassInfo_SD
 method GetClass_SD            (  ) : ClassInfo_SD
 method GetAttNum              ( i : INDEX ) : ATTNUM
 method GetCard                ( i : INDEX ) : CARD
 method GetClassName           (  ) : Str
 method GetID                  (  ) : OBJECT_ID
 method GetNumAttVals          (  ) : INDEX
 method GetNumAttributes       (  ) : ATTNUM
 method GetNumLinks            (  ) : LINKNUM
 method GetAttnameByIndex      ( i : INDEX ) : Str
 method GetIndValByIndex       ( i : INDEX) : Str
 method GetNumLinkVals         ( ) : INDEX
 method GetLinkValByIndex_SD   ( i : INDEX ) : Editable_SD
 method IsNoneLinkByIndex      ( i : INDEX ) : BOOL
 method GetLinkNameByIndex     ( i : INDEX) : Str
 method GetObjectName          (  ) : Str
 method GetLinkCard            ( L : LINKNUM ) : CARD
 method GetAttCard             (A : ATTNUM) : CARD
 method IsUnknownLink          ( L : LINKNUM, C : CARD) : BOOL
 method IsNoneLink             ( L : LINKNUM, C : CARD ) : BOOL
 method IsIAO                  (  ) : BOOL
 method EGetLink_SD            ( L : LINKNUM,
                                 C : CARD ) : Editable_SD

 // continued ...
```

```
... // continued

 method GetAtt                 ( A : ATTNUM,
                                 C : CARD ) : Str
 method GetIndVal              ( A : ATTNUM,
                                 C : CARD ) : Str
 method SetAtt                 ( A : ATTNUM,
                                 C : CARD ) : Str
 method IsUnknownAtt           ( A : ATTNUM,
                                 C : CARD ) : BOOL
 method SetAtt                 ( val : Str,
                                 A : ATTNUM,
                                 C : CARD)  : ErrorResultType
 method IsErrorToSetAtt        ( val : Str,
                                 A : ATTNUM,
                                 C : CARD ) : ErrorResultType
 method AddAttVal              ( val : Str,
                                 A : ATTNUM,
                                 C : CARD ) : ErrorResultType
 method IsErrorToAddAttVal     ( val : Str,
                                 A : ATTNUM,
                                 C : CARD)  : ErrorResultType
 method RemoveAttVal           ( A : ATTNUM,
                                 C : CARD ) : ErrorResultType
 method IsErrorToRemoveAttVal ( A : ATTNUM, C : CARD ): ErrorResultType
 method RemoveUnknownAttVals  ( A : ATTNUM ) : ErrorResultType
 method SetLink                ( linkVal : Editable_SD,
                                 L : LINKNUM,
                                 C : CARD ) : BOOL
 method SetLink                ( val : Str,
                                 L : LINKNUM,
                                 C : CARD ) : ErrorResultType
 method IsErrorToSetLink       ( val : Str,
                                 L : LINKNUM,
                                 C : CARD) : ErrorResultType
 method AddLinkVal             ( val : Str,
                                 L : LINKNUM,
                                 C : CARD ) : ErrorResultType
 method IsErrorToAddLinkVal    ( val : Str,
                                 L : LINKNUM,
                                 C : CARD ) : ErrorResultType
 method RemoveLinkVal          ( L : LINKNUM,
                                 C : CARD ) : ErrorResultType
 method IsErrorToRemoveLinkVal( L : LINKNUM,
                                  C : CARD ) : ErrorResultType
 method RemoveUnknownAndNoneLinkVals(L : LINKNUM) : ErrorResultType
 method Copy                   ( source : Editable_SD) : ErrorResultType
 method RecalcAllLiveRoutes    (  )
 method GetDeletePredictor     ( L : LINKNUM,
                                 C : CARD ) : DLlistEdPtr_SD
 method GetDeletePredictor     (  ) : DLlistEdPtr_SD
 method FindFirstLinkNum       ( destination : Editable_SD ) : LINKNUM
 method FindFirstLinkCard      ( destination : Editable_SD ) : CARD
endclass
```

```
class ClassInfo_SD isa
 method IsKindOf_SD            ( parent : ClassInfo_SD ) : BOOL
 method GetLinkName            ( L : LINKNUM ) :  Str
 method GetAttributeName       ( A : ATTNUM ) : Str
 method GetLinkCard            ( L : LINKNUM ) : CARD
 method GetAttCard             ( A : ATTNUM ) : CARD
endclass
```

```
class Output_SD isa
 method GetName                (  ) : Str
 method SetGoals               ( listVals : DLlistStr,
                                 listProbs : DLlistReal,
                                 costOrBenefit : OutputType  )
 method GetGoalsValues         (  ) : DLlistStr
 method GetGoalsProbs          (  ) : DLlistReal
 method GetGoalsType           (  ) : OutputType
 method GetAchievedValues      ( ) : DLlistStr
 method GetAchievedProbs       (  ) : DLlistReal
 method GetTarget_SD           (  ) : UTarget_SD
 method GetNumGoalVals         (  ) : INDEX
 method GetNumAchievedVals     (  ) : INDEX
 method FGetGoalVal            ( i : INDEX ) : FLOAT
 method FGetAchievedVal        ( i : INDEX ) : FLOAT
 method BuildAuditTrail        (  ) : DLlistStr
 method GetAuditPars           (  ) : AuditPars
 method SetAuditPars           ( auditParameters : AuditPars )
endclass
```

```
class UTarget_SD isa
 method IsCts                  (  ) : BOOL
 method IsPointValue           (  ) : BOOL
 method GetHistPars            (  ) : HistPars
 method Generate               ( histogramParameters : HistPars )
 method GetDDEStringData       (  ) : Str
 method GetDDEStringName       (  ) : Str
 method GetCorrelationCoeff    ( otherUTarget : Utarget,
                                 cor_type : CorrelationType ) : FLOAT
 method GetMax                 (  ) : Str
 method GetMin                 (  ) : Str
 method GetSD                  (  ) : Str
 method GetSkew                (  ) : str
 method GetKurt                (  ) : Str
 method GetMean                (  ) : Str
 method GetPercentiles         ( percentageBegin : FLOAT,
                                 numberPercentiles : INTEGER,
                                 percentageInterval : FLOAT ) : DLlistStr
 method GetSampleSize          (  ) : INDEX
 method FGetSample             ( sampleIndex : INDEX ) : FLOAT
 method IGetSample             ( sampleIndex : INDEX ) : INTEGER
 method GetRoundedMax          ( numberBars : INDEX ) : FLOAT
 method GetRoundedMin          ( numberBars : INDEX ) : FLOAT
 method IGetDiscreteValue      ( discreteValueIndex : INDEX ) : INTEGER
 method GetBarHeights          (  ) : ArrayFloat
endclass
```

Appendix L: Comparison Between rand() and Shuffling Table Method

In this appendix the performance of two alternative pseudo-random number generators are compared, when used to evaluate an example risk model in RiTo. The first generator, *rand()* is supplied with the compiler and assumed to consist of a single linear congruential generator (LCG). The second, *UniformRand()* is an implementation of the algorithm using three LCGs and a shuffling table described in Section 9.1.3 of Chapter 9. It was noted in that section that, whilst the shuffling table method is considerably more computationally expensive than *rand()*, it avoids the sequential correlations displayed by a simple LCG. This appendix discusses whether either of these distinctions are significant when the random number generators are used in conjunction with a Latin hypercube sampling strategy, and the risk model evaluation algorithm, on a typical model containing 36 random variables.

The significance of the computational expense is considered simply by comparing the total simulation time required using each of the two generators. The significance of the sequential correlations is measured by observing the sample correlation coefficient between pairs of random variables using each generator and then comparing the correlation coefficient values thus obtained. Since the measured correlation coefficient values are subject to sampling error, repeated simulations are performed and the mean correlation coefficient value is determined for each method, along with the standard deviation in the difference between the correlation coefficient values. Using this information, we can perform a significance test on the hypothesis that:

"the average value of (ρs - ρr) is zero"

where

ρs = the correlation coefficient obtained using the shuffling table method
ρr = the correlation coefficient obtained using *rand()*.

The example risk model used to conduct the comparison was Case 5 of the Rover case study, described in Section 11.2.7 of Chapter 11. Samples were generated for ***a*** = `InteriorTrim.piece_cost` and ***b*** = `Generic FloorCoverings.piece_cost` using each of the two methods and the correlation coefficient between the samples for ***a*** and ***b*** was evaluated. This was repeated using 30 different seed values for the generators. All simulations contained 1000 runs, and the time taken for each simulation (on a 120 MHz Pentium) lay between 3 and 4 seconds whether the shuffling table method was used or rand().

The results obtained were as follows:

$\mu diff$ = mean value of (ρs - ρr) = `0.00236`
$\sigma diff$ = standard deviation in value of (ρs - ρr) = `0.02415`

The test-statistic, z, is given by

$$z = \frac{\mu diff - 0}{\sigma diff / \sqrt{30}}$$

and thus takes a value of `0.53462`. Thus we cannot reject the hypothesis at the 5% significance level, since the test statistic lies within the range [-2.04, +2.04] (which gives the rejection region for a t-distribution with 30 degrees of freedom in a two-tailed test).

It can be concluded that, in this example, there is no evidence of a significant difference in the correlation coefficient values achieved using the *rand()* generator from those achieved using the shuffling table method. Similarly, the greater computation time required for the shuffling table method is insignificant in comparison with the time expended on activities other than pseudo-random number generation during a Monte Carlo simulation of a risk model.